11-12-1996

Demodulation of Narrowband Radio Frequency Signals by Aliasing Sampling

Chun-Ching Lin
Portland State University

Recommended Citation

Lin, Chun-Ching, "Demodulation of Narrowband Radio Frequency Signals by Aliasing Sampling" (1996). *Dissertations and Theses.* Paper 5286.

10.15760/etd.7159

THESIS APPROVAL

The abstract and thesis of Chun-Ching Lin for the Master of Science in Electrical Engineering were presented November 12, 1996, and accepted by the thesis committee and the department.

COMMITTEE APPROVALS:

[redacted]
Y. C. Jenq, Chair

[redacted]
Branimir Pejcinovic

[redacted]
Eugene Enneking
Representative of the Office of Graduate Studies

DEPARTMENT APPROVAL:

[redacted]
Rolf Schaumann, Chair
Department of Electrical Engineering

* *

ACCEPTED FOR PORTLAND STATE UNIVERSITY BY THE LIBRARY

by [redacted] on 20 November 1996

ABSTRACT

An abstract of the thesis of Chun-Ching Lin for the Master of Science in Electrical Engineering presented November 12, 1996.

Title: Demodulation of Narrowband Radio Frequency Signals by Aliasing Sampling

The objective of this thesis is to study the demodulation of narrowband radio frequency signals by aliasing sampling in order to reduce the sampling rate. The spectrum can be recreated at the lower frequency position by aliasing sampling. However, if the sampling rate is deviated from the desired one, error will occur. The sensitivity to the frequency error of aliasing sampling is studied.

One main reason of the deviation of the sampling rate is the frequency drifting of the local oscillator. Being able to compensate the oscillator drifting errors inexpensively, automatic frequency control (AFC) loops are important at receivers. Two major digital AFC algorithms are studied. One is the Phase method AFC, and the other is the Magnitude method AFC. Study indicates that both methods perform almost equally well. One adaptive AFC algorithm is also proposed. The scheme of the adaptive AFC algorithm is to use Upper-bound and Lower-bound techniques to squeeze the frequency errors. It is shown that the adaptive AFC algorithm can achieve

up to 20 dB average signal-to-noise power ratio over the Magnitude method AFC under a noisy environment.

DEMODULATION OF NARROWBAND RADIO FREQUENCY SIGNALS BY ALIASING SAMPLING

by

CHUN-CHING LIN

A thesis submitted in partial fulfillment of
requirements for the degree of

MASTER OF SCIENCE
in
ELECTRICAL ENGINEERING

Portland State University
1996

ACKNOWLEDGMENT

I would like to thank my advisor, Professor Y. C. Jenq, for his knowledge and the members of my committee for their support. I would also like to thank Shirley for her continuous encouragement during my study at Portland State University. Finally, without the assistance from my parents, this project can not be accomplished at all.

TABLE OF CONTENTS

PAGE

ACKNOWLEDGMENT ---------- ii

LIST OF TABLES ---------- vi

LIST OF FIGURES ---------- vii

CHAPTER

I INTRODUCTION ---------- 1

Demodulation by Aliasing Sampling ---------- 1

Automatic Frequency Control Algorithms ---------- 4

II SENSITIVITY TO THE FREQUENCY ERROR OF ALIASING SAMPLING ---------- 7

Introduction ---------- 7

Frequency Error Prediction ---------- 9

Summary ---------- 14

III DIGITAL AUTOMATIC FREQUENCY CONTROL ALGORITHMS ---------- 15

Introduction ---------- 15

Phase Method AFC Algorithm ---------- 16

Configuration
With Noise
Performances With Noise

Magnitude Method AFC Algorithm ---------- 24

Configuration
With Noise
Performances With Noise

Summary ----------32

IV NOISE REDUCTION BY FILTERING

Introduction ---------- 34

The Phase Method AFC With Filters ----------35

The Magnitude Method AFC With Filters ----------38

Summary ----------43

V ADAPTIVE AUTOMATIC FREQUENCY CONTROL ALGORITHM ---------- 44

Introduction ---------- 44

Notations ----------45

Ranges of the Parameters ----------46

Initial Conditions ---------- 47

Program ---------- 47

Determine the Next Local Frequency Value
Adjust *Hit_Up* Counter, *Hit_Low* Counter, and *Hit_None* Counter
Adjust the Next *Step_Margin*
Move the Next *Upper_Bound*
Move the Next *Lower_Bound*

Effects of the Parameters ----------51

Performances at the Steady State
Performances at the Transient State

Summary ----------57

VI CONCLUSIONS ----------58

REFERENCES ----------60

APPENDIX ----------61

LIST OF TABLES

TABLE		PAGE
I	The Performances of the Phase Method AFC	24
II	The Performances of the Magnitude Method AFC	32
III	The Performances of the Phase Method AFC With Filters	37
IV	The Performances of the Magnitude Method AFC With Filters	39
V	The Performances of the Magnitude Method AFC With Two Averaging Processes	42
VI	Different Adaptive AFC Configurations	51
VII	The Performances of the Adaptive AFC Algorithm	54

LIST OF FIGURES

FIGURE PAGE

1. Using aliasing sampling to demodulate a narrowband signal by creating an image of the spectrum at the baseband ----------3

2. The configuration of an AFC loop ----------4

3. The demodulation by aliasing sampling when the sampling rate is deviated ---------- 7

4. The baseband signal (Top) and its Fourier transform (Bottom) ----- 8

5. ESRs at the different sampling rates ---------- 10

6 ESRs at the different sampling rates under the same carrier frequency ---------- 11

7. ESRs at the different carrier frequencies under the same sampling rate ---------- 12

8. The ESR against the sampling rate in terms of the frequency deviation percentage ---------- 13

9. The configuration of the Phase method AFC ----------17

10. The phasor subtraction of two complex samples in the Phase method AFC without noise ----------19

11. The phasor subtraction of two complex samples in the Phase method AFC with noise ----------21

12. The performances of the Phase method AFC with noise ---------- 23

13. The configuration of the Magnitude method AFC ----------25

14. The magnitudes of the DFT of two complex samples in the Magnitude method AFC. (Top) Without noise.

(Bottom) With noise. ----------26

15. The performances of the Magnitude method AFC ----------31

16. Noise reduction by using averaging filters in the Phase method AFC ----------36

17. Noise reduction by using averaging filters in the Magnitude method AFC ----------38

18. The performances of the Magnitude method AFC with filters by averaging the predictions and by averaging the corresponding magnitudes ----------41

19. The comparison of the performances of both the Phase method AFC and the Magnitude method AFC with the Filter 1, 2, and 3, respectively ---------- 43

20. The flowchart for the adaptive AFC algorithm ----------50

21. The performances of the adaptive AFC algorithm ----------53

22. The performances of the adaptive AFC algorithm if there is a step change of the carrier frequency. (Top) of the first configuration. (Bottom) of the third configuration ---------- 56

CHAPTER I

INTRODUCTION

DEMODULATION BY ALIASING SAMPLING

In the world of communications, there are broad usages of radio frequency signals. For instance, amplitude modulation broadcast, television, cellular mobile radio, and satellite are some examples of application of radio frequency signals. In order to increase the efficiency of transmission, the baseband signal is modulated by a high frequency carrier. After the modulated waveform is received by the receiver, the goal of demodulation is to shift the spectrum to the final baseband signal frequency. This spectrum shifting can be accomplished by either analog or digital methods.

In the analog demodulation process, the spectrum is shifted to an intermediate frequency first. This translation in frequency is called superheterodyne [1]. The accomplishment of superheterodyne is the multiplication of a modulated waveform by a locally generated sinusoidal signal. Since the oscillator's performances [2] are affected by temperature and power variation, superheterodyne could be inaccurate, and therefore the performances of the analog receivers will be degraded.

Due to the flexibility of digital signal processing techniques, the frequency translation can also be accomplished by the digital methods mentioned earlier.

According to the sampling theorem [1], the sampling rate has to be higher than twice the bandwidth of the signal in order to reconstruct the signal faithfully. In other words, the sampling rate has to be high enough to sample the modulated waveform and to accomplish superheterodyne digitally. Because a low sampling rate is more cost effective than a high sampling rate, one way to reduce the sampling rate and achieve superheterodyne digitally is aliasing sampling if the modulated waveform is a narrowband signal. As the sampling rate is chosen properly, superheterodyne is performed digitally by producing each identical image of the spectrum at low frequency positions. Also, the spectrum of a narrowband signal can be shifted down to the baseband by using aliasing sampling. In other words, an image of the narrowband signal can be created at 0 Hz by a digital demodulation, if the assumed sampling rate is an integer submultiple of the carrier frequency [3]. One demonstration of the demodulation of a narrowband signal by aliasing sampling is shown in Figure 1.

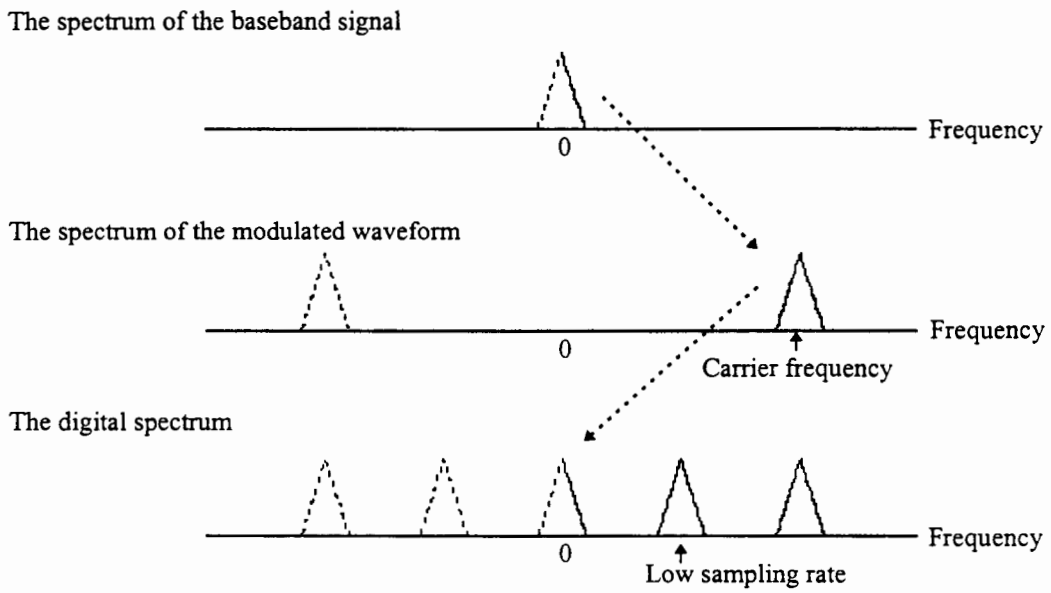

Figure 1. Using aliasing sampling to demodulate a narrowband signal by creating an image of the spectrum at the baseband.

Therefore, aliasing sampling is a solution to avoid the high sampling rate as well as to accomplish superheterodyne digitally in the demodulation of narrowband radio frequency signals with high frequency carriers.

Since the sampling clock is driven by a local oscillator, the accuracy of the sampling rate is governed by the oscillator's performances. If the sampling rate is not as accurate as expected, every image of the spectrum will deviate from its desired frequency position proportionally, i.e., the digital spectrum distortion will occur. Therefore, the accuracy of the sampling rate is important in a digital demodulation. In the first part of this thesis, the sensitivity to the frequency error of aliasing sampling will be investigated.

AUTOMATIC FREQUENCY CONTROL ALGORITHMS

How to maintain the stability and accuracy of oscillators is a practical issue in analog and digital demodulations. In other words, to capture the carrier frequency with respect to the local oscillator's frequency is important in the process of demodulation. One solution is to use the high precision oscillators to perform the task at receivers. However, high precision also means high cost and the frequency errors may be from the carrier because of the noise. A more economical method is to replace oscillators of high cost by the AFC loops [4] at receivers.

To compensate oscillator drifting errors, an AFC loop at a receiver is to detect the carrier. The function of an AFC loop is to produce a local sinusoidal signal with the same frequency as the carrier frequency. This sinusoidal signal can be sent to perform either superheterodyne in an analog demodulation, or to drive a sampling clock in a digital demodulation. Therefore, one AFC loop can operate either in an analog or a digital demodulation.

The configuration of an AFC loop [4] is shown in Figure 2.

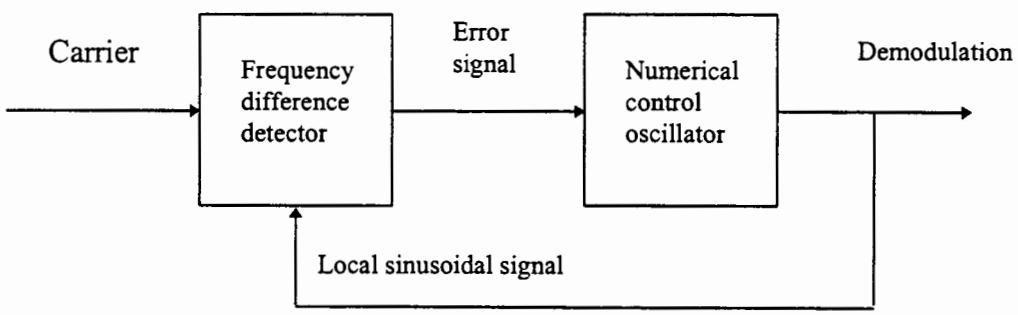

Figure 2. The configuration of an AFC loop.

The basic components of an AFC loop are a frequency difference detector and a numerical control oscillator (NCO). The input of an AFC loop is a carrier which is extracted from the modulated waveform. The frequency error between the carrier and the local oscillator can be converted into an error signal by the frequency difference detector. According to this error signal, the NCO generates a new local sinusoidal signal as a feedback to the frequency difference detector. Meanwhile this local sinusoidal signal of the carrier frequency is the output of the AFC loop. Although the oscillator drifting can occur at the transmitter or at the receiver, the frequency errors can be reduced by the AFC loop.

There are different digital AFC loops. To approach the real-time digital AFC processing, two digital AFC loops of simple algorithm will be studied in the second part, and in the third part, of this thesis, respectively. One is the Phase method AFC [5]. The other is the Magnitude method AFC which is a modified version of the discrete Fourier transform (DFT) AFC [6]. In the Phase method AFC, the carrier frequency is determined by the phase difference of two consecutive samples, while in the Magnitude method AFC, the carrier frequency is determined by the magnitude square difference at two discrete frequencies. It is found that both methods are fast but sensitive to noise. In order to improve the noise immunity of two digital AFC methods, averaging filters will be used to obtain the new estimates of the desired frequency difference between the carrier and the local oscillator. In the fourth part of

this thesis, the performances of these two methods using averaging filters will be presented.

Finally, an adaptive AFC algorithm will be proposed. The scheme of the adaptive AFC algorithm is to use Upper-bound and Lower-bound techniques to squeeze the frequency error. In the last part of this thesis, the adaptive AFC algorithm will be studied and its performances will be presented.

THESIS ORGANIZATION

The organization of this thesis is as follow. The sensitivity to the frequency error of aliasing sampling is investigated in Chapter II. The Phase method AFC is described and its performances are presented in Chapter III. The Magnitude method AFC is described and its performances are presented in Chapter IV. The adaptive AFC algorithm is proposed and its performances are presented in Chapter V. The conclusion is given in Chapter VI.

CHAPTER II

SENSITIVITY TO THE FREQUENCY ERROR OF ALIASING SAMPLING

INTRODUCTION

The spectrum shifting in the process of demodulation can be achieved by aliasing sampling. However, if the sampling rate is deviated from the desired one, the digital spectrum distortion will occur. One demonstration of the digital spectrum distortion is shown in Figure 3.

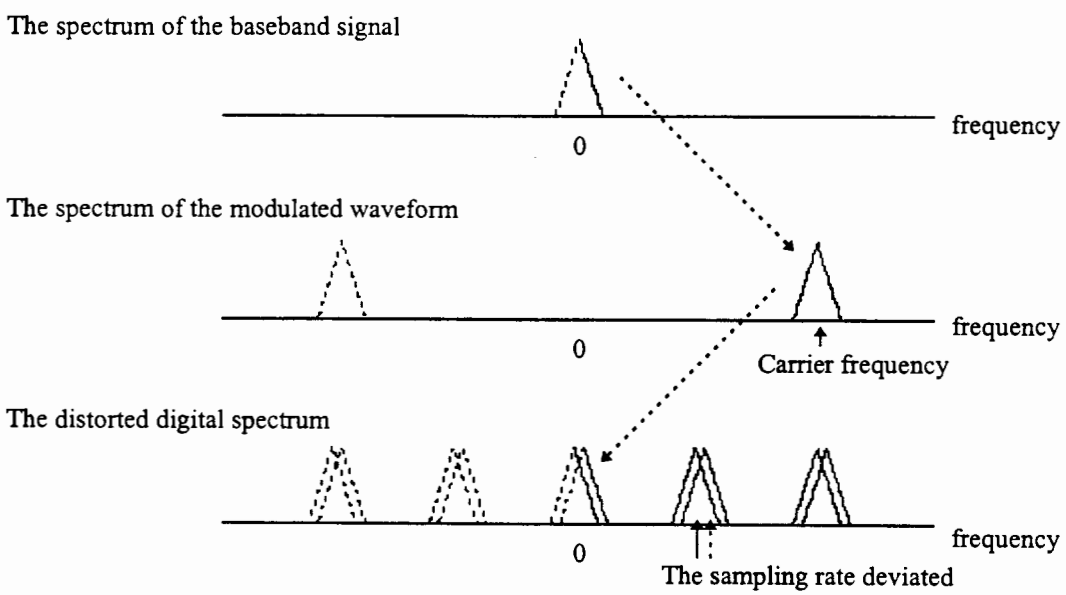

Figure 3. The demodulation by aliasing sampling when the sampling rate is deviated.

The objective of this chapter is to study the sensitivity to the frequency error of aliasing sampling. It is difficult to approach this objective without having the baseband signal and the modulated waveform. The assumptions are made as follow. Let the baseband signal be $f(t) = \frac{\sin^2 \pi t}{\pi^2 t^2}$. The baseband signal and its Fourier transform [1] are shown in Figure 4.

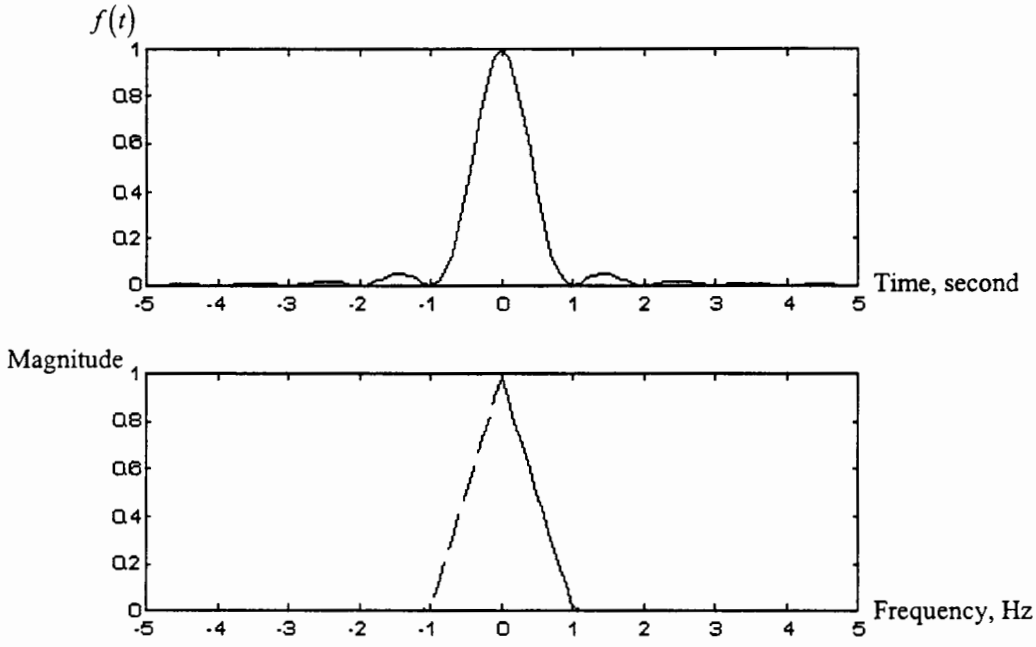

Figure 4. The baseband signal (Top) and its Fourier transform (Bottom).

The bandwidth of the baseband signal is 1 Hz. Let the modulated waveform of an amplitude modulation system [1] be $f(t) \cdot \cos(2\pi f_C t)$. Let the carrier frequency f_C be 20 Hz in order to consider the error in concept directly. Let the sampling rate f_S

be higher than twice the bandwidth of the baseband signal to meet the Nyquist criterion [1], i.e., $f_S > 2$ Hz.

FREQUENCY ERROR PREDICTION

Suppose that the error is from the sampling rate only. In order to measure the sensitivity to the frequency error of aliasing sampling, the mean-square error to signal power ratio (ESR) is evaluated at the different sampling rates. The ESR is expressed as the following equation.

$$ESR = \frac{\sum_n \left(x[n] - x[n]' \right)^2}{\sum_n x[n]^2} \times 100\%$$

where $x[n]$ is obtained from sampling the baseband signal and $x[n]'$ is obtained from samples of the modulated waveform, respectively. The demodulation of the modulated waveform by aliasing sampling is simulated and measured. The ESR against the sampling rate is shown in Figure 5.

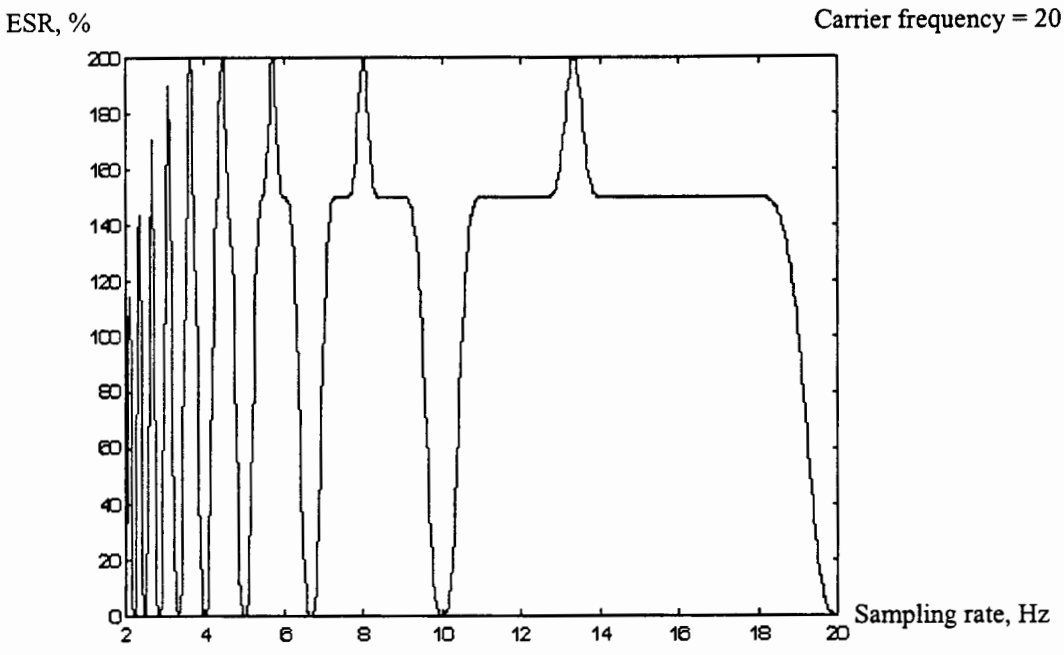

Figure 5. ESRs at the different sampling rates.

In order to recover the baseband signal from the modulated waveform of an amplitude modulation system by aliasing sampling, an image of the spectrum is recreated at the baseband. Ideally, the ESR is 0% when one image of the spectrum is created by aliasing sampling at 0 Hz position exactly. In other words, the ESR is 0% when the sampling rate is an integer submultiple of the carrier frequency [3] as expressed in the following equation.

$$f_S = \frac{f_C}{N}$$
$$= \frac{20}{1}, \frac{20}{2}, \frac{20}{3}, \cdots > 2$$

where N is a positive integer such that f_S meets the Nyquist criterion.

However, if the ESR is too high, it is an indication that the image of the spectrum is far away from the baseband. Assume that the sampling rate is deviated from the desired one such that the ESR is less than 100 %. As shown in Figure 5 before, the ESRs for the various sampling rates under the same carrier frequency are shown in Figure 6.

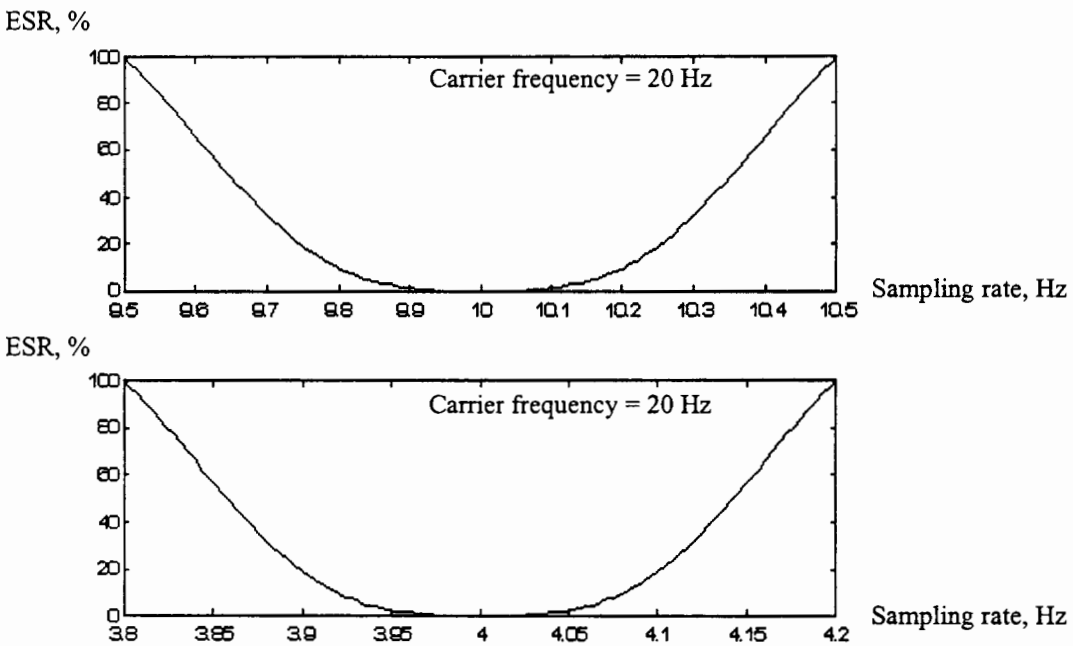

Figure 6. ESRs at the different sampling rates under the same carrier frequency.

It is observed that the ESR curves in Figure 6 are identical. If the carrier frequency of the modulated waveform is set to be 200 Hz and 2000 Hz, respectively, the demodulation of the modulation waveform by aliasing sampling is simulated and measured. The ESRs for the various carrier frequencies under the same sampling rate are shown in Figure 7.

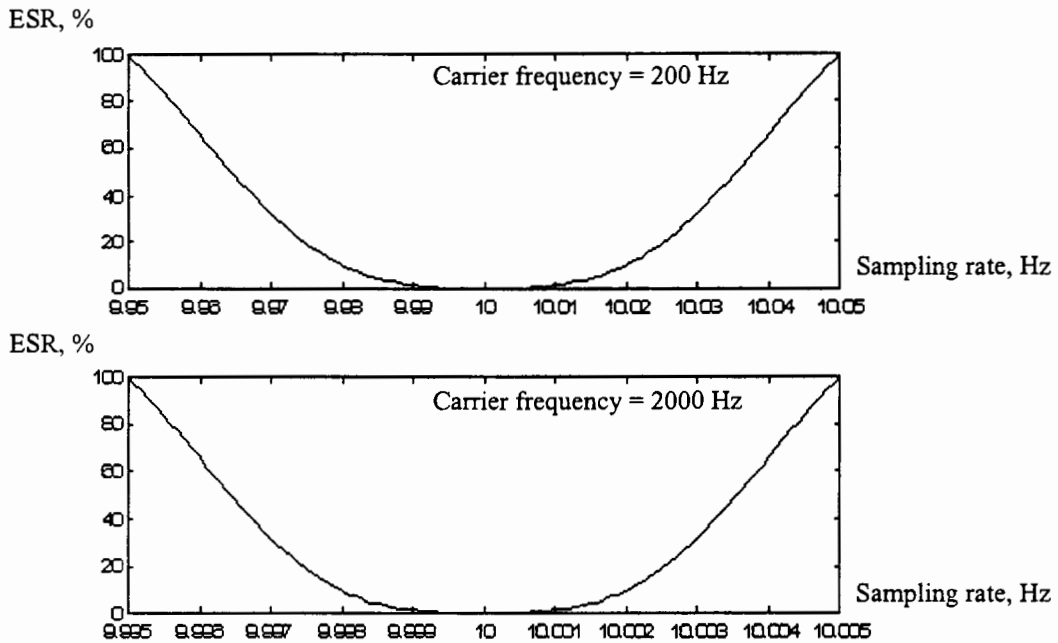

Figure 7. ESRs at the different carrier frequencies under the same sampling rate.

It is observed that all ESR curves in Figure 6 and Figure 7 are identical. If the ESR is plotted against the sampling rate in terms of the frequency deviation percentage, there will be just one curve. Let f_P denote the sampling rate in terms of the frequency deviation percentage as expressed in the following equation.

$$f_P = \frac{f_S - f_D}{\frac{f_D}{f_C}} \times 100\%$$

where f_D is one of $\frac{f_C}{N}$ closest to the sampling rate. The ESR against the sampling rate in terms of the frequency deviation percentage is shown in Figure 8.

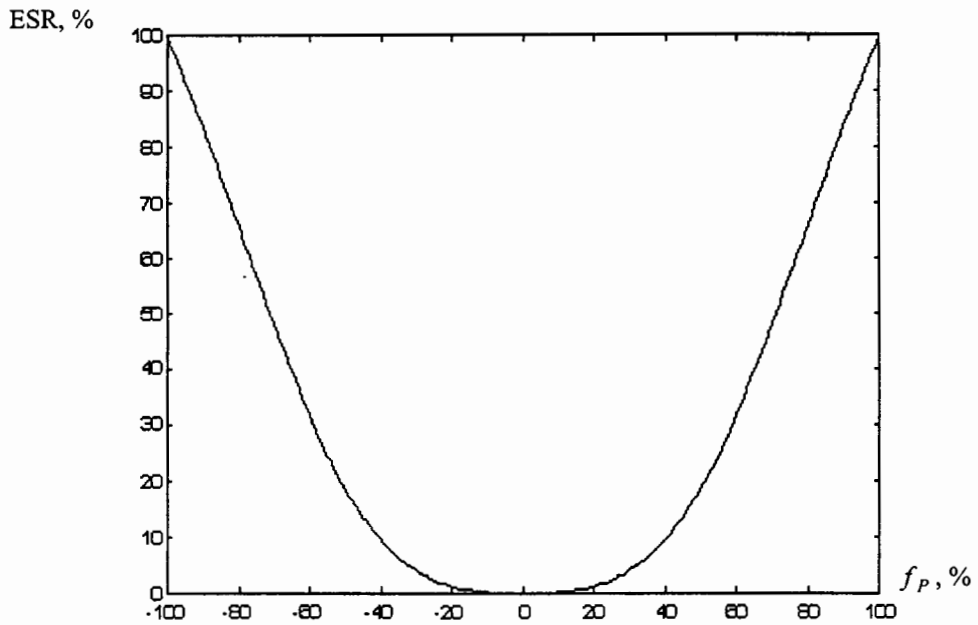

Figure 8. The ESR against the sampling rate in terms of the frequency deviation percentage.

By finding a formula of the curve in Figure 8 via the polynomial curve fitting, the ESR will be predictable. Therefore, the ESR is higher than 100% if $|f_S - f_D| > \frac{f_D}{f_C}$.

Otherwise, the ESR is approximated according to the following equation.

$$ESR = C_6|f_P|^6 + C_5|f_P|^5 + C_4|f_P|^4 + C_3|f_P|^3 + C_2|f_P|^2 + C_1|f_P|$$

where $C_6 = 4.6862$

$C_5 = -14.0520$

$C_4 = 13.5126$

$C_3 = -4.1003$

$C_2 = 1.0298$

$C_1 = -0.0726$

The error of the polynomial curve fitting of the ESR is found to be less than 0.2 %.

SUMMARY

When the sampling rate is an integer submultiple of the carrier frequency, the spectrum of a narrowband signal can be recreated at the baseband. The ESR is investigated under the assumption of the known baseband signal $f(t) = \frac{\sin^2 \pi t}{\pi^2 t^2}$ in an amplitude modulation system. The contributions in this chapter are to investigate the sensitivity to the frequency error of aliasing sampling as follow. The ESR is more sensitive to the sampling rate deviated from the desired low sampling rate than to the one deviated from the desired high sampling rate as shown in Figure 6. The ESR is more sensitive to the deviated sampling rate at the high carrier frequency than to the deviated one at the low carrier frequency as shown in Figure 7. Also, the other contribution in this chapter is the prediction of the ESR as follow. The ESR lower than 100% can be predicted by fitting the ESR curve in terms of the frequency deviation percentage as shown in Figure 8.

CHAPTER III

DIGITAL AUTOMATIC FREQUENCY CONTROL ALGORITHMS

INTRODUCTION

In the process of an analog or digital demodulation, the amount of the spectrum shifting in frequency depends on the frequency difference between the carrier and the local oscillator. Because the oscillator drifting could occur at the transmitter or at the receiver or both, the AFC loop [4] can be used to compensate the uncertainty of the frequency errors. In this chapter, algorithms of the Phase method AFC [5] and the Magnitude method AFC will be described. The performances of two methods with noise will be also evaluated. Before the configurations are examined, the following assumptions apply to both methods.

The input of an AFC loop is a single frequency carrier $s(t)$ with an initial phase θ.

$$s(t) = \cos(2\pi f_C t + \theta) + A_N(t)\cos(2\pi f_C t + \phi(t)) \qquad (3.1)$$

where the noise amplitude $A_N(t)$ is of the Rayleigh distribution, and the noise phase $\phi(t)$ is uniformly distributed between 0 and 2π [7]. The Rayleigh density function is expressed as follow.

$$p_x(x) = \begin{cases} \frac{x}{\alpha^2} e^{\frac{-x^2}{2\alpha^2}} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

The input carrier frequency f_C is assumed to be located in a pre-designated frequency interval $[f_{LOW},\ f_{HIGH}]$. This frequency interval is called the carrier band. The bandwidth of the carrier band is $(f_{HIGH} - f_{LOW})$. Let f_L be the local oscillator's frequency (local frequency), and an initial value of f_L can be set to be within that carrier band. The frequency deviation is $\Delta f = f_L - f_C$.

PHASE METHOD AFC ALGORITHM

There are three sections to describe the Phase method AFC algorithm and evaluate its performances with noise.

Configuration

Without consideration of noise at the moment, the input carrier is $s(t) = \cos(2\pi f_C t + \theta)$, i.e., $A_N(t) = 0$ in Equation (3.1). The Phase method AFC is to detect Δf by the phase difference between two samples from a complex sinusoidal signal of Δf Hz. The configuration is shown in Figure 9.

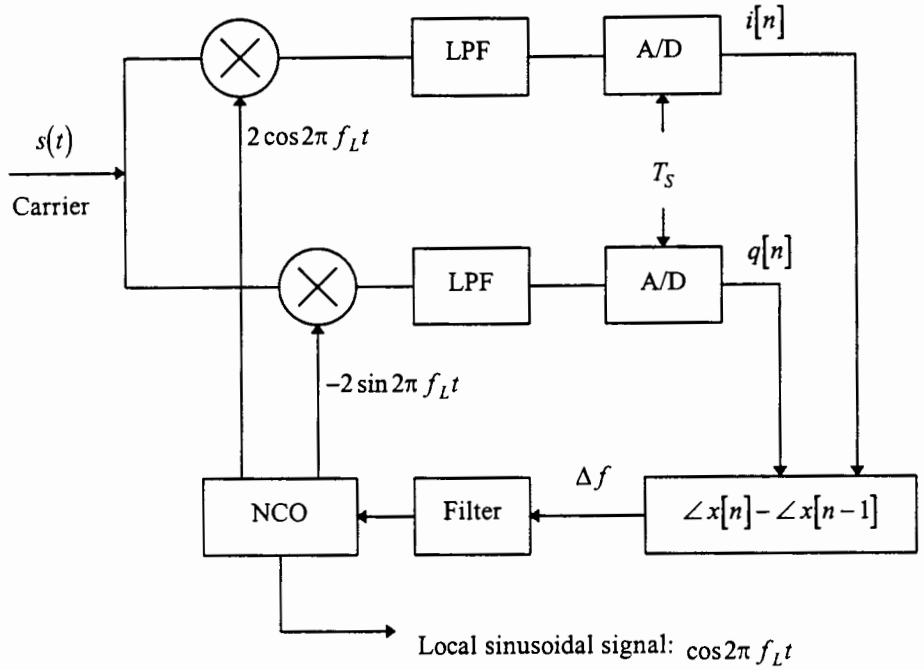

Figure 9. The configuration of the Phase method AFC.

The two mixers will produce the in-phase and the quadrature components of the input carrier. The outputs of the mixers are both frequency sum components and frequency difference components between the carrier and the local sinusoidal signal. The two low pass filters have the same bandwidth equal to the largest possible deviation frequency $|\Delta f|$. The reasonable bandwidth of the low pass filter has to be at least the bandwidth of the carrier band. The low pass filters will reject frequency sum components and pass frequency difference components. The in-phase component $i(t)$ becomes the real part, and the quadrature component $q(t)$ becomes the imaginary part of a complex signal, respectively. Therefore, a complex sinusoidal signal $x(t)$ of Δf Hz is created at the baseband.

$$\begin{aligned} x(t) &= i(t) + jq(t) \\ &= \cos(2\pi \Delta f t + \theta) + j\sin(2\pi \Delta f t + \theta) \\ &= e^{j(2\pi \Delta f t + \theta)} \end{aligned} \tag{3.2}$$

In order to extract Δf from Equation (3.2), two consecutive samples, $x[0] = x((n-1)T_S)$ and $x[1] = x(nT_S)$, are taken from $x(t)$ at the sampling period of T_S second. The complex sample $x[n]$ can be thought a phasor. The phase angle $\angle x[n]$, of the phasor $x[n]$, is expressed as follow.

$$\angle x[n] = \begin{cases} \tan^{-1}\dfrac{\mathrm{Im}(x[n])}{\mathrm{Re}(x[n])} + \pi & \text{if } \mathrm{Re}(x[n]) < 0 \text{ and } \mathrm{Im}(x[n]) > 0 \\ \tan^{-1}\dfrac{\mathrm{Im}(x[n])}{\mathrm{Re}(x[n])} - \pi & \text{if } \mathrm{Re}(x[n]) < 0 \text{ and } \mathrm{Im}(x[n]) < 0 \\ \tan^{-1}\dfrac{\mathrm{Im}(x[n])}{\mathrm{Re}(x[n])} & \text{otherwise} \end{cases}$$

The phase difference between these two samples is

$$\begin{aligned} \angle x[1] - \angle x[0] &= (2\pi \Delta f n T_S + \theta) - (2\pi \Delta f (n-1) T_S + \theta) \\ &= 2\pi \Delta f T_S \end{aligned} \tag{3.3}$$

The phasor subtraction of these two complex samples is shown in Figure 10.

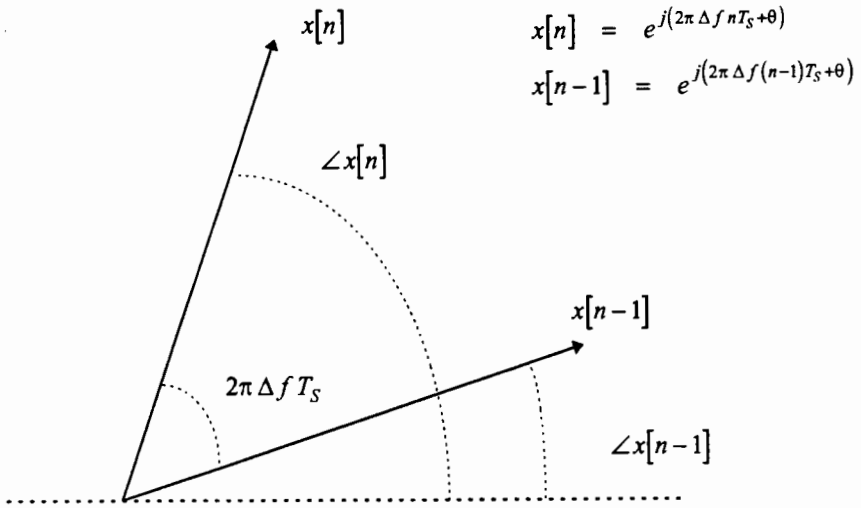

Figure 10. The phasor subtraction of two complex samples in the Phase method AFC without noise.

In order to obtain the correct Δf from Equation (3.3), two requirements need to be addressed. First, $(\angle x[1]-\angle x[0])$ must be the principal value of phase. Since the phase range of the subtraction of two phasors, $(\angle x[1]-\angle x[0])$, is $[-2\pi,\ 2\pi]$, the result of Equation (3.3) is modified according to the following equation.

$$\angle x[1]-\angle x[0] = \begin{cases} \angle x[1]-\angle x[0]-2\pi & \text{if } \angle x[1]-\angle x[0]>\pi \\ \angle x[1]-\angle x[0]+2\pi & \text{if } \angle x[1]-\angle x[0]<-\pi \\ \angle x[1]-\angle x[0] & \text{otherwise} \end{cases}$$

Secondly, T_S has to meet the Nyquist criterion. Since the local frequency could be higher or lower than the carrier frequency, the extreme values of Δf could be positive or negative value of the bandwidth of the carrier band, i.e., $-(f_{HIGH}-f_{LOW})<\Delta f<(f_{HIGH}-f_{LOW})$, and hence the Nyquist criterion is $1/T_S>2\cdot(f_{HIGH}-f_{LOW})$.

Therefore, Δf can be obtained according to the following equation.

$$\Delta f = \frac{\angle x[1] - \angle x[0]}{2\pi T_S} \tag{3.4}$$

where $(\angle x[1] - \angle x[0])$ is the principal value of phase and $1/T_S > 2 \cdot (f_{HIGH} - f_{LOW})$.

After the frequency deviation is detected, Δf is sent through a filter and to the NCO. The function of the filter is to reduce the effect of the noise, and it will be discussed in the next chapter. Finally, the next local frequency is set to be $(f_L - \Delta f)$.

So far one iteration of the Phase method AFC has been completed. Ideally, the carrier frequency can be grabbed by using two complex samples in the Phase method AFC. Now, let us consider the situation of the input carrier with noise.

With noise

After the input carrier, $s(t)$ in Equation (3.1), going through the mixer and the low pass filters, the complex sinusoidal signal with noise $\hat{x}(t)$ is generated at the baseband. The quantity $\hat{x}(t)$ can be expressed as follow.

$$\hat{x}(t) = e^{j(2\pi \Delta f t + \theta)} + A_N(t) e^{j(2\pi \Delta f t + \phi(t))} \tag{3.5}$$

Two consecutive samples, $\hat{x}[0] = \hat{x}((n-1)T_S)$ and $\hat{x}[1] = \hat{x}(nT_S)$, taken from $\hat{x}(t)$ are used to obtain an estimate, $\Delta \hat{f}$, of Δf. The phase angle $\angle \hat{x}[n]$, of the phasor $\hat{x}[n]$, is expressed as follow.

$$\angle\hat{x}[n] = \begin{cases} \tan^{-1}\dfrac{\mathrm{Im}(\hat{x}[n])}{\mathrm{Re}(\hat{x}[n])} + \pi & \text{if } \mathrm{Re}(\hat{x}[n]) < 0 \text{ and } \mathrm{Im}(\hat{x}[n]) > 0 \\ \tan^{-1}\dfrac{\mathrm{Im}(\hat{x}[n])}{\mathrm{Re}(\hat{x}[n])} - \pi & \text{if } \mathrm{Re}(\hat{x}[n]) < 0 \text{ and } \mathrm{Im}(\hat{x}[n]) < 0 \\ \tan^{-1}\dfrac{\mathrm{Im}(\hat{x}[n])}{\mathrm{Re}(\hat{x}[n])} & \text{otherwise} \end{cases}$$

The phasor subtraction of two complex samples with noise is shown in Figure 11.

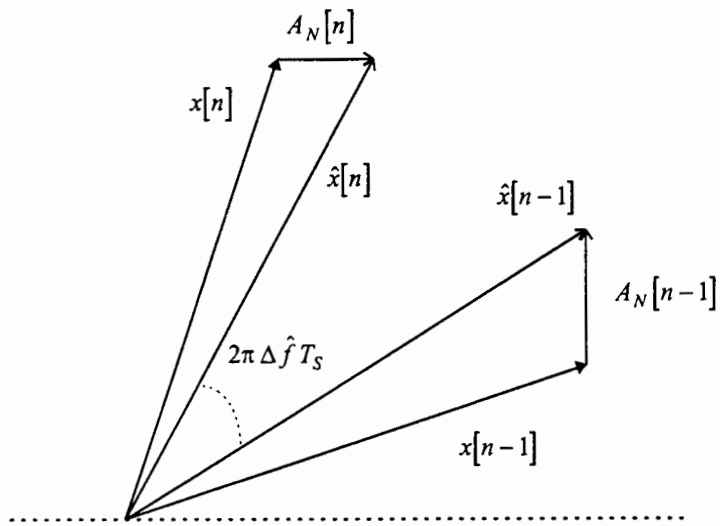

Figure 11. The phasor subtraction of two complex samples in the Phase method AFC with noise.

In order to obtain an estimate, $\Delta\hat{f}$, of Δf similar to Equation (3.4), two requirements need to be addressed. First, the phase range of the subtraction of two phasors, $(\angle\hat{x}[1] - \angle\hat{x}[0])$ is modified according to the following equation.

$$\angle\hat{x}[1] - \angle\hat{x}[0] = \begin{cases} \angle\hat{x}[1] - \angle\hat{x}[0] - 2\pi & \text{if } \angle\hat{x}[1] - \angle\hat{x}[0] > \pi \\ \angle\hat{x}[1] - \angle\hat{x}[0] + 2\pi & \text{if } \angle\hat{x}[1] - \angle\hat{x}[0] < -\pi \\ \angle\hat{x}[1] - \angle\hat{x}[0] & \text{otherwise} \end{cases}$$

Secondly, T_S has to meet the Nyquist criterion. Therefore, similar to Equation (3.4), $\Delta\hat{f}$ may be estimated according to the following equation.

$$\Delta\hat{f} = \frac{\angle\hat{x}[1]-\angle\hat{x}[0]}{2\pi T_S} \tag{3.6}$$

where $\left(\angle\hat{x}[1]-\angle\hat{x}[0]\right)$ is the principal value of phase and $1/T_S > 2\cdot\left(f_{HIGH}-f_{LOW}\right)$. Finally, the next local frequency may be estimated by $\left(f_L-\Delta\hat{f}\right)$.

Performances with noise

To investigate the performances of the Phase method AFC with noise, the root mean-square (RMS) values of Δf are simulated and measured at different SNRs. The SNR is expressed as follow.

$$SNR = 10\log\frac{signal\ power}{noise\ power}$$
$$= 10\log\frac{\frac{1}{2}}{\sigma^2}$$

where σ^2 is the variance of $A_N(t)$. From the visual inspection, the RMS frequency errors are measured at the steady state and the data are large enough to ignore the statistic error. The initial conditions of the simulations are as follow. The frequency interval of the carrier band is from 97.5 MHz to 102.5 MHz. The desired carrier frequency is 100 MHz. The initial local frequency is 98 MHz. The Nyquist rate is 2(102.5-97.5)=10 MHz. The sampling rate is 25 MHz. The steady state RMS errors

as the percentages of the bandwidth of the carrier band at the different SNRs are shown in Figure 12.

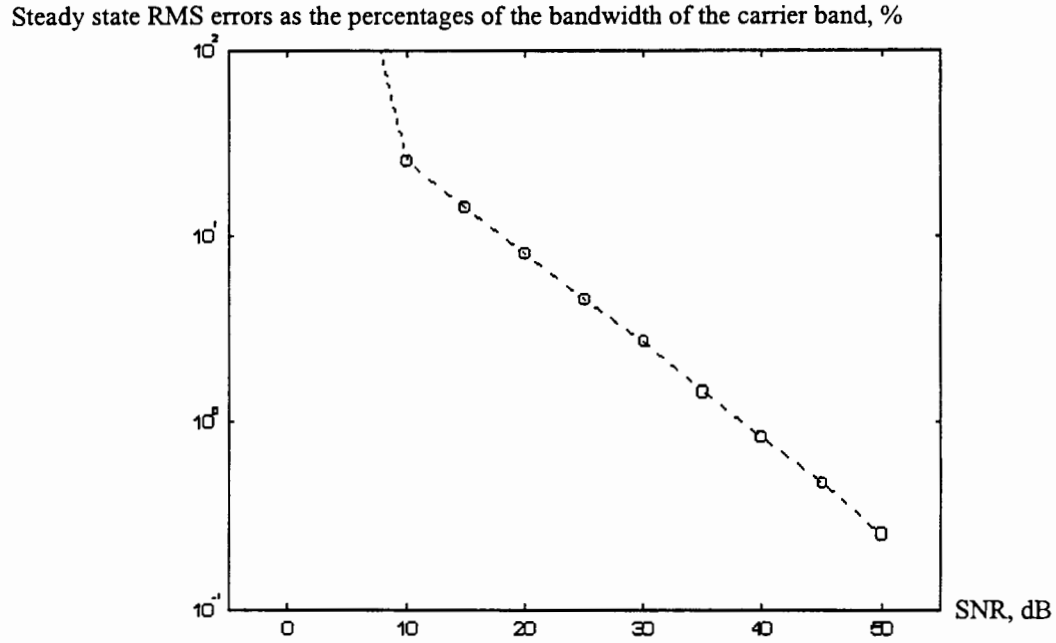

Figure 12. The performances of the Phase method AFC with noise.

The data are shown in Table I.

TABLE I

THE PERFORMANCES OF THE PHASE METHOD AFC

SNR	Steady state RMS error as the percentage of the bandwidth of the carrier band
0 dB	> 100%
5 dB	> 100%
10 dB	25.0925%
15 dB	14.1378%
20 dB	8.1047%
25 dB	4.6140%
30 dB	2.7408%
35 dB	1.4383%
40 dB	0.8265%
45 dB	0.4742%
50 dB	0.2499%

It is seen that when the SNRs are less than 10 dB, the RMS error percentages are close to 100%, i.e., the local frequency can not converge within the carrier band. It is also noted that, at SNR = 35 dB, Δf is about 1% of the bandwidth of the carrier band.

MAGNITUDE METHOD AFC ALGORITHM

There are three sections to describe the Magnitude method AFC algorithm and evaluate its performances with noise.

Configuration

Without consideration of the noise at the moment, the Magnitude method AFC is to detect Δf by the magnitude square difference at two discrete frequencies in a digital spectrum from sampling a complex sinusoidal signal of Δf Hz. The configuration is shown in Figure 13.

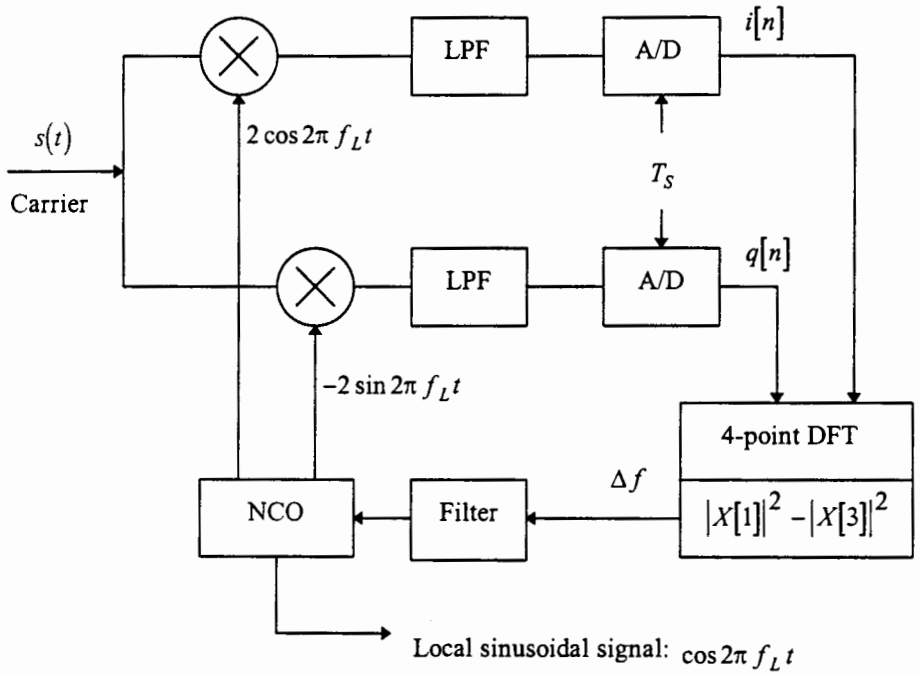

Figure 13. The configuration of the Magnitude method AFC.

The generation of that complex sinusoidal signal is the same as the one in the Phase method AFC. After the input carrier $s(t)$ passing through the mixers and the low pass filters, the complex sinusoidal signal $x(t)$ is the same as the one from Equation (3.2).

$$x(t) = e^{j(2\pi \Delta f t + \theta)}$$

Two consecutive samples, taken from $x(t)$ at the sampling period of T_S second, and two padding zeros form a 4-point sequence $x[n]$.

$$\begin{aligned} x[0] &= x\big((n-1)T_S\big) = e^{j(2\pi\Delta f(n-1)T_S+\theta)} \\ x[1] &= x(nT_S) = e^{j(2\pi\Delta f nT_S+\theta)} \\ x[2] &= 0 \\ x[3] &= 0 \end{aligned}$$

The DFT of $x[n]$ is the four complex samples $X[k]$ in the frequency domain [1].

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}kn}$$

where the length of the sequence N is 4, and the index k goes from 0 to 3. The magnitudes of the 4-point DFT without noise are shown in Figure 14 (Top).

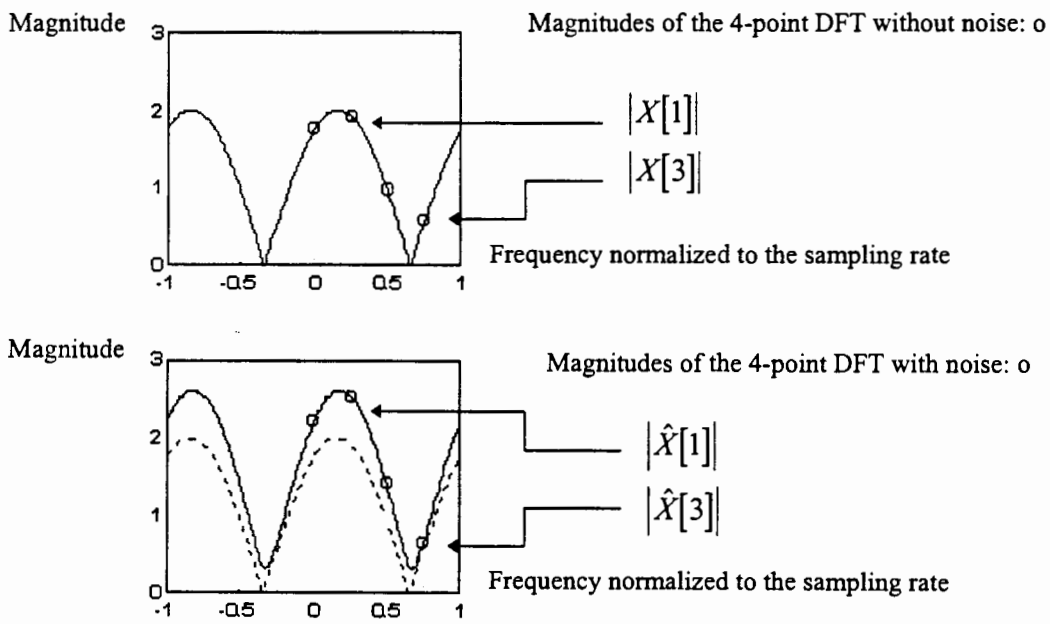

Figure 14. The magnitudes of the DFT of two complex samples in the Magnitude method AFC. (Top) Without noise. (Bottom) With noise.

The magnitude square of $X[k]$ is given by

$$|X[k]|^2 = 2 + 2\cos\left(2\pi \Delta f T_S - \frac{\pi}{2}k\right) \tag{3.7}$$

The Δf can be extracted from the difference between $|X[1]|^2$ and $|X[3]|^2$.

$$\begin{aligned} |X[1]|^2 - |X[3]|^2 &= \left(2 + 2\cos\left(2\pi \Delta f T_S - \frac{\pi}{2}\cdot 1\right)\right) - \left(2 + \cos\left(2\pi \Delta f T_S - \frac{\pi}{2}\cdot 3\right)\right) \\ &= 4\sin 2\pi \Delta f T_S . \end{aligned}$$

Therefore,

$$2\pi \Delta f T_S = \sin^{-1}\frac{1}{4}\left(|X[1]|^2 - |X[3]|^2\right) \tag{3.8}$$

In order to obtain the correct Δf from Equation (3.8), two requirements need to be addressed. First, since the range of $|X[k]|^2$ in Equation (3.7) is [0, 4], the range of $\frac{1}{4}\left(|X[1]|^2 - |X[3]|^2\right)$ in Equation (3.8) is [-1, 1].

$$\frac{1}{4}\left(|X[1]|^2 - |X[3]|^2\right) = \begin{cases} 1 & \text{if } \frac{1}{4}\left(|X[1]|^2 - |X[3]|^2\right) > 1 \\ -1 & \text{if } \frac{1}{4}\left(|X[1]|^2 - |X[3]|^2\right) < -1 \\ \frac{1}{4}\left(|X[1]|^2 - |X[3]|^2\right) & \text{otherwise} \end{cases}$$

This is because that when the possible value of $\frac{1}{4}\left(|X[1]|^2 - |X[3]|^2\right)$ exceeds its range, it is an indication that the magnitude spectrum must have been heavily corrupted by the noise. Secondly, the sampling rate $1/T_S$ decides the range of Δf. From Equation

(3.8), the range of the function $\sin^{-1}$ is $\left[\frac{-\pi}{2}, \frac{\pi}{2}\right]$, and hence $\frac{-\pi}{2} < 2\pi \Delta f T_S < \frac{\pi}{2}$, and the range of Δf is $\left[-\frac{1}{4T_S}, \frac{1}{4T_S}\right]$. Since the maximum of $|\Delta f|$ has to cover the bandwidth of the carrier band, the sampling rate $1/T_S$ in the Magnitude method AFC has to be 4 times larger than the bandwidth of the carrier band. Therefore, Δf is obtained according to the following equation.

$$\Delta f = \frac{1}{2\pi T_S} \sin^{-1} \frac{1}{4}\left(|X[1]|^2 - |X[3]|^2\right) \qquad (3.9)$$

where the range of $\frac{1}{4}\left(|X[1]|^2 - |X[3]|^2\right)$ is [-1, 1], and $1/T_S > 4 \cdot (f_{HIGH} - f_{LOW})$. Finally, the next local frequency is set to be $(f_L - \Delta f)$. The function of the filter will be discussed in the next chapter.

So far one iteration of the Magnitude method AFC has been completed. Similar to the Phase method AFC, the Magnitude method AFC can capture the carrier frequency by using two complex samples and two padding zeros. Now, let us consider the situation of the input carrier with noise in the Magnitude method AFC.

With noise

The generation of the complex sinusoidal signal with noise is the same as the one in the Phase method AFC. From Equation (3.5), the complex sinusoidal signal with noise is

$$\hat{x}(t) = e^{j(2\pi\Delta f t+\theta)} + A_N(t)e^{j(2\pi\Delta f t+\phi(t))}$$

Two consecutive samples, $\hat{x}\big((n-1)T_S\big)$ and $\hat{x}(nT_S)$, taken from $\hat{x}(t)$ at the sampling period T_S second, and two padding zeros form the 4-point sequence $\hat{x}[n]$.

$$\begin{aligned}\hat{x}[0] &= \hat{x}\big((n-1)T_S\big)\\ &= e^{j(2\pi\Delta f(n-1)T_S+\theta)} + A_N\big((n-1)T_S\big)e^{j\left(2\pi\Delta f(n-1)T_S+\phi\left((n-1)T_S\right)\right)}\\ &= e^{j(2\pi\Delta f(n-1)T_S+\theta)} + A_N[0]e^{j(2\pi\Delta f(n-1)T_S+\phi[0])}\end{aligned}$$

where $A_N[0] = A_N\big((n-1)T_S\big)$ and $\phi[0] = \phi\big((n-1)T_S\big)$.

$$\begin{aligned}\hat{x}[1] &= \hat{x}(nT_S)\\ &= e^{j(2\pi\Delta f nT_S+\theta)} + A_N(nT_S)e^{j(2\pi\Delta f nT_S+\phi(nT_S))}\\ &= e^{j(2\pi\Delta f nT_S+\theta)} + A_N[1]e^{j(2\pi\Delta f nT_S+\phi[1])}\end{aligned}$$

where $A_N[1] = A_N(nT_S)$ and $\phi[1] = \phi(nT_S)$.

$$\hat{x}[2] = 0$$

$$\hat{x}[3] = 0$$

The magnitudes of the 4-point DFT with noise are shown in Figure 14 (Bottom). The magnitude square difference between $\left|\hat{X}[1]\right|^2$ and $\left|\hat{X}[3]\right|^2$ can be used to obtain an estimate, $\Delta\hat{f}$, of Δf.

$$\begin{aligned}\frac{1}{4}\left(\left|\hat{X}[1]\right|^2 - \left|\hat{X}[3]\right|^2\right) &= \sin 2\pi\Delta f T_S\\ &\quad + A_N[0]\sin\left(2\pi\Delta f T_S + \theta - \phi[0]\right)\\ &\quad + A_N[1]\sin\left(2\pi\Delta f T_S + \phi[1] - \theta\right)\\ &\quad + A_N[0]A_N[1]\sin\left(2\pi\Delta f T_S + \phi[1] - \phi[0]\right) \qquad (3.10)\end{aligned}$$

Two requirements of obtaining an estimate, $\Delta\hat{f}$, of Δf need to be addressed. First, the range of $\frac{1}{4}\left(\left|\hat{X}[1]\right|^2 - \left|\hat{X}[3]\right|^2\right)$ is modified according to the following equation.

$$\frac{1}{4}\left(\left|\hat{X}[1]\right|^2 - \left|\hat{X}[3]\right|^2\right) = \begin{cases} 1 & \text{if } \frac{1}{4}\left(\left|\hat{X}[1]\right|^2 - \left|\hat{X}[3]\right|^2\right) > 1 \\ -1 & \text{if } \frac{1}{4}\left(\left|\hat{X}[1]\right|^2 - \left|\hat{X}[3]\right|^2\right) < -1 \\ \frac{1}{4}\left(\left|\hat{X}[1]\right|^2 - \left|\hat{X}[3]\right|^2\right) & \text{otherwise} \end{cases}$$

Secondly, the sampling rate has to be 4 times higher than the bandwidth of the carrier band. Therefore, similar to Equation (3.9), $\Delta\hat{f}$ may be estimated according to the following equation.

$$\Delta\hat{f} = \frac{1}{2\pi T_S}\sin^{-1}\frac{1}{4}\left(\left|\hat{X}[1]\right|^2 - \left|\hat{X}[3]\right|^2\right) \qquad (3.11)$$

where the range of $\frac{1}{4}\left(\left|\hat{X}[1]\right|^2 - \left|\hat{X}[3]\right|^2\right)$ is [-1, 1], and $1/T_S > 4\cdot\left(f_{HIGH} - f_{LOW}\right)$. Finally, the next local frequency my be estimated by $\left(f_L - \Delta\hat{f}\right)$.

Performances with noise

To investigate the performances of the Magnitude method AFC with noise, the steady state RMS errors are simulated and measured at different SNRs. The initial conditions of the simulations are the same as the ones in the Phase method AFC. The frequency interval of the carrier band is from 97.5 MHz to 102.5 MHz. The desired carrier frequency is 100 MHz. The initial local frequency is 98 MHz. The lowest

sampling rate has to be 4(102.5-97.5)=20 MHz. The sampling rate used in the simulation is 25 MHz. The steady state RMS errors as the percentages of the bandwidth of the carrier band at the different SNRs are shown in Figure 15.

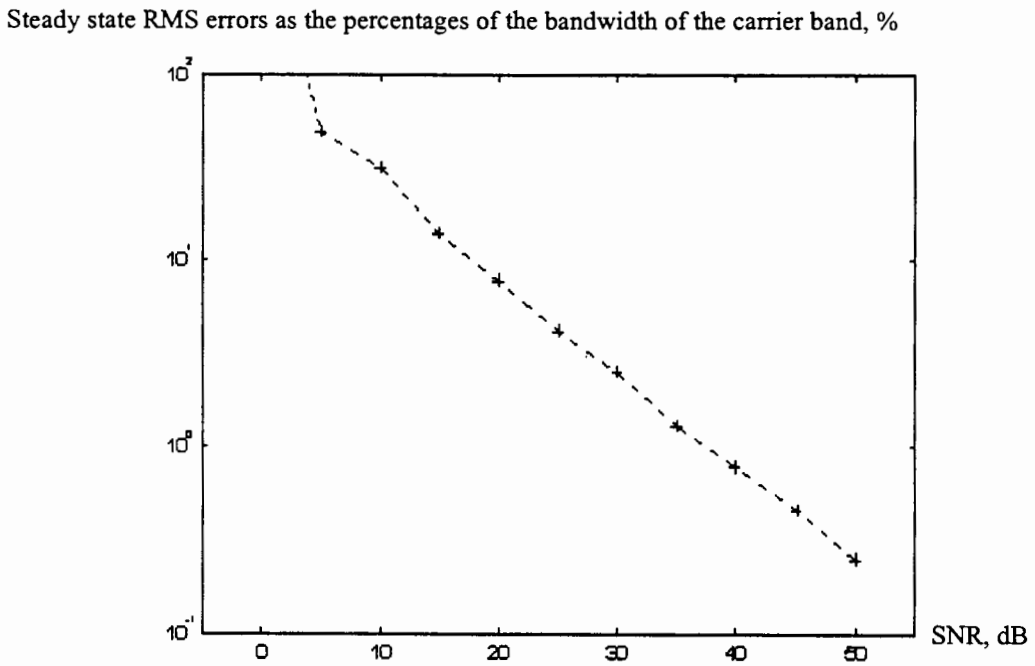

Figure 15. The performances of the Magnitude method AFC.

The data are shown in Table II.

TABLE II

THE PERFORMANCES OF THE MAGNITUDE METHOD AFC

SNR	Steady state RMS error as the percentage of the bandwidth of the carrier band
0 dB	> 100%
5 dB	59.8514%
10 dB	27.1312%
15 dB	15.0025%
20 dB	8.5342%
25 dB	4.6997%
30 dB	2.6607%
35 dB	1.5327%
40 dB	0.8103%
45 dB	0.4845%
50 dB	0.2581%

It is seen that when the SNRs are less than 5 dB, the RMS error percentages are close to 100 %, i.e. the local frequency can not converge within the carrier band. It is also noted that, at SNR = 35 dB, Δf is about 1% of the bandwidth of the carrier band.

SUMMARY

In this chapter, two digital AFC algorithms have been examined and their performances with noise are presented. The contributions in this chapter are to investigate the performances of the Phase method AFC and propose the Magnitude method AFC. At high SNRs, both methods perform almost equally well. At low

SNRs, for example, at 5 dB, the steady state RMS errors of the Phase method AFC are larger than those of the Magnitude method AFC. In this regard, the Magnitude method AFC performs better than the Phase method AFC does. Theoretically, both digital AFC algorithms can grab the carrier frequency by using two samples only. Study indicates that the RMS errors are large at low SNR. In the following chapter, the averaging filters are used to improve the noise immunity of two digital AFC methods.

CHAPTER IV

NOISE REDUCTION BY FILTERING

INTRODUCTION

To minimize the noise impact upon both AFC methods, averaging filters are used to produce a new estimate of Δf. The NCO will not adjust the local frequency until the new estimate of Δf is available. There are three averaging filters of different length in the simulations. Those filters are called Filter 1, Filter 2, and Filter 3. The Filter 1 will produce a new estimate of Δf from averaging the previous 10 $\Delta \hat{f}$. The Filter 2 will produce a new estimate of Δf from averaging the previous 100 $\Delta \hat{f}$. The Filter 3 will produce a new estimate of Δf from averaging the previous 1000 $\Delta \hat{f}$. In order to predict how the SNR is improved by the averaging filter, the following assumptions are made.

Suppose that $Y_1, Y_2, \cdots, Y_L$ are random variables representing L consecutive estimates of Δf made by a digital AFC method. Assume that $Y_1, Y_2, \cdots, Y_L$ are independent variables and each of them is with the same mean and the same variance σ^2. Suppose that $\overline{Y}$ is a random variable representing the output of an averaging filter.

$$\begin{aligned}\overline{Y} &= \frac{1}{L}(Y_1 + Y_2 + \cdots + Y_L) \\ &= \frac{1}{L}Y_1 + \frac{1}{L}Y_2 + \cdots + \frac{1}{L}Y_L\end{aligned}$$

where L is the length of the averaging filter. The variance of $\overline{Y}$ is obtained according to the following equation.

$$\begin{aligned}Var(\overline{Y}) &= \left(\frac{1}{L}\right)^2 Var(Y_1) + \left(\frac{1}{L}\right)^2 Var(Y_2) + \cdots + \left(\frac{1}{L}\right)^2 Var(Y_L) \\ &= \left(\frac{1}{L}\right)^2 \sigma^2 + \left(\frac{1}{L}\right)^2 \sigma^2 + \cdots + \left(\frac{1}{L}\right)^2 \sigma^2 \\ &= \frac{\sigma^2}{L}\end{aligned}$$

The SNR improvement by using the averaging filter is

$$\begin{aligned}&SNR_{With\ Filter} - SNR_{Without\ Filter} \\ &= 10\log\frac{signal\ power}{\frac{\sigma^2}{L}} - 10\log\frac{signal\ power}{\sigma^2} \qquad (4.1) \\ &= 10\log L \quad (dB)\end{aligned}$$

Therefore, the expectations of the SNR improvements by using the Filter 1, the Filter 2, and the Filter 3 are $10\log 10 = 10\,(dB)$, $10\log 100 = 20\,(dB)$, and $10\log 1000 = 30\,(dB)$, respectively.

THE PHASE METHOD AFC WITH FILTERS

In the Phase method AFC [5], $\frac{\angle\hat{x}[1]-\angle\hat{x}[0]}{2\pi T_S}$ in Equation (3.6) is the sample used in the averaging filter. The averaging filter will average all the samples and obtain a new estimate of Δf according to Equation (3.6). The initial conditions of our simulation are the same as the one without filter in the Phase method AFC. The steady state RMS errors as the percentage of the bandwidth of the carrier band at the Filter 1, the Filter 2, and the Filter 3 are shown in Figure 16.

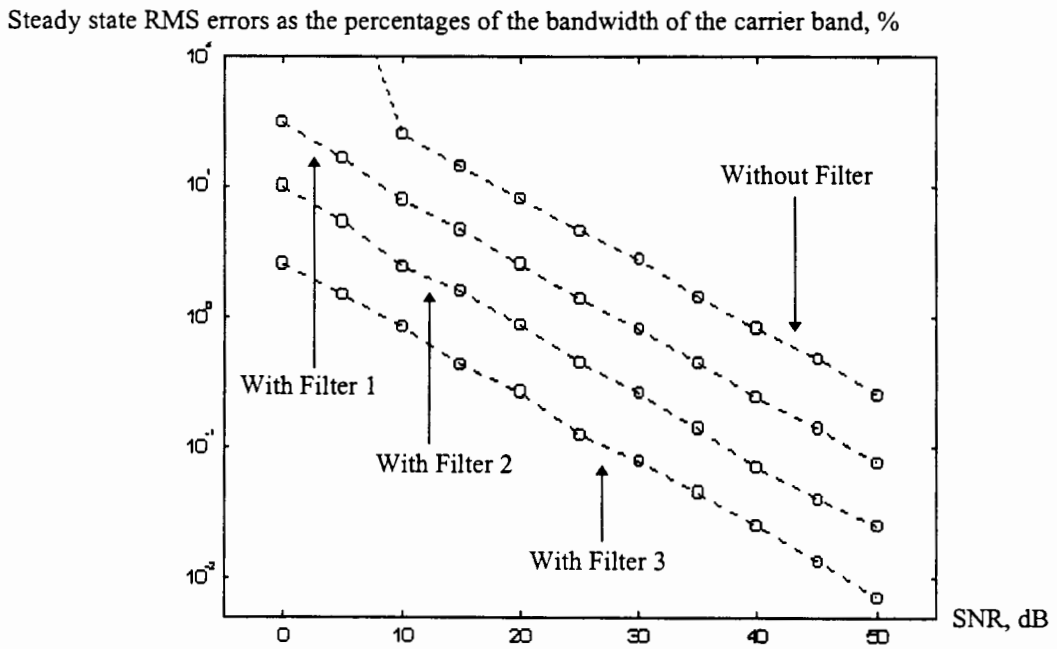

Figure 16. Noise reduction by using averaging filters in the Phase method AFC.

The data are shown in Table III.

TABLE III

THE PERFORMANCES OF THE PHASE METHOD AFC WITH FILTERS

	Steady state RMS error as the percentage of the bandwidth of the carrier band			
SNR	No Filter	Filter 1	Filter 2	Filter 3
0 dB	> 100%	30.9186%	10.2179%	2.5540%
5 dB	> 100%	16.2892%	5.3829%	1.4732%
10 dB	25.0925%	7.9828%	2.3822%	0.8434%
15 dB	14.1378%	4.6453%	1.5882%	0.4261%
20 dB	8.1047%	2.5782%	0.8607%	0.2665%
25 dB	4.6140%	1.3970%	0.4440%	0.1275%
30 dB	2.7408%	0.7918%	0.2616%	0.0781%
35 dB	1.4383%	0.4546%	0.1414%	0.0457%
40 dB	0.8265%	0.2454%	0.0700%	0.0253%
45 dB	0.4742%	0.1420%	0.0407%	0.0137%
50 dB	0.2499%	0.0769%	0.0253%	0.0072%
SNR improvement expected by the different filters	0 dB	10 dB	20 dB	30 dB
Number of iterations before calculating the RMS error	100	50	10	10
Number of iterations during calculating the RMS error	1000	500	100	100

The approximate SNR improvements in the Phase method AFC by using the Filter 1, the Filter 2, and the Filter 3 are 10 dB, 20 dB and 30 dB, respectively. Three

averaging filters of different length meet the expectation of the SNR improvement in Equation (4.1).

THE MAGNITUDE METHOD AFC WITH FILTERS

In the Magnitude method AFC, from Equation (3.10), $\frac{1}{4}\left(\left|\hat{X}[1]\right|^2 - \left|\hat{X}[3]\right|^2\right)$ with the modified range is the sample used in the averaging filter. The averaging filter will average all the samples and obtain a new estimate of Δf according to Equation (3.11). The steady state RMS errors as the percentage of the bandwidth of the carrier band at the Filter 1, the Filter 2, and the Filter 3 are shown in Figure 17.

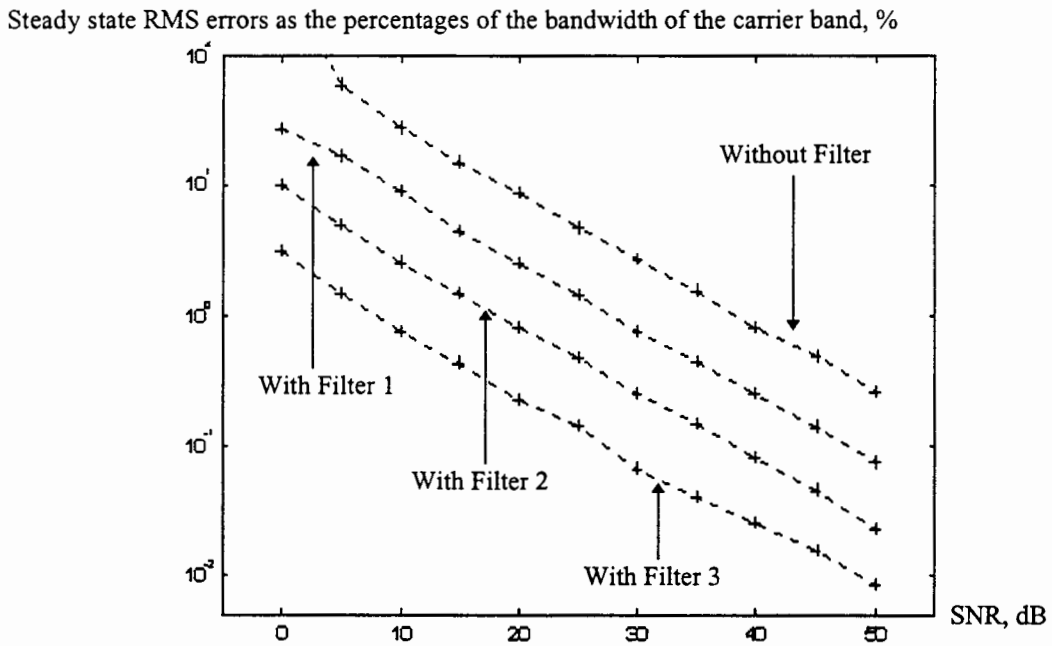

Figure 17. Noise reduction by using averaging filters in the Magnitude method AFC.

The data are shown in Table IV.

TABLE IV

THE PERFORMANCES OF THE MAGNITUDE METHOD AFC WITH FILTERS

	Steady state RMS error as the percentage of the bandwidth of the carrier band			
SNR	No Filter	Filter 1	Filter 2	Filter 3
0 dB	> 100%	26.7059%	9.7584%	3.0848%
5 dB	59.8514%	16.8008%	4.9791%	1.4665%
10 dB	27.1312%	9.0833%	2.5390%	0.7593%
15 dB	15.0025%	4.4860%	1.4677%	0.4318%
20 dB	8.5342 %	2.4952%	0.8050%	0.2282%
25 dB	4.6997%	1.4225%	0.4792%	0.1432%
30 dB	2.6007%	0.7576%	0.2492%	0.0670%
35 dB	1.5327%	0.4500%	0.1477%	0.0412%
40 dB	0.8103%	0.2515%	0.0802%	0.0251%
45 dB	0.4845%	0.1395%	0.0453%	0.0154%
50 dB	0.2581%	0.0745%	0.0229%	0.0083%
SNR improvement expected by the different filters	0 dB	10 dB	20 dB	30 dB
Number of iterations before calculating the RMS error	100	50	10	10
Number of iterations during calculating the RMS error	1000	500	100	100

The approximate SNR improvements in the Magnitude method AFC by using the Filter 1, the Filter 2, and the Filter 3 are 10 dB, 20 dB and 30 dB, respectively. Three averaging filters of different length meet the expectation of the SNR improvement in Equation (4.1). One may wonder why the averaging filter averages the corresponding magnitudes rather than does every estimate of Δf from the Magnitude method AFC directly. This is because when the averaging filter averages the corresponding magnitudes in Equation (3.10), the error of the estimate of Δf will not be introduced until the operation of $\sin^{-1}$ in Equation (3.11) is performed. However, when the averaging filter averages every estimation from the Magnitude method AFC in Equation (3.11) directly, the error of the estimate of Δf is introduced once after one iteration of the Magnitude method AFC is completed. Therefore, the RMS errors of averaging magnitudes are smaller than the ones of averaging estimates at low SNR, and both are almost the same at high SNR. The results are shown in Figure 18.

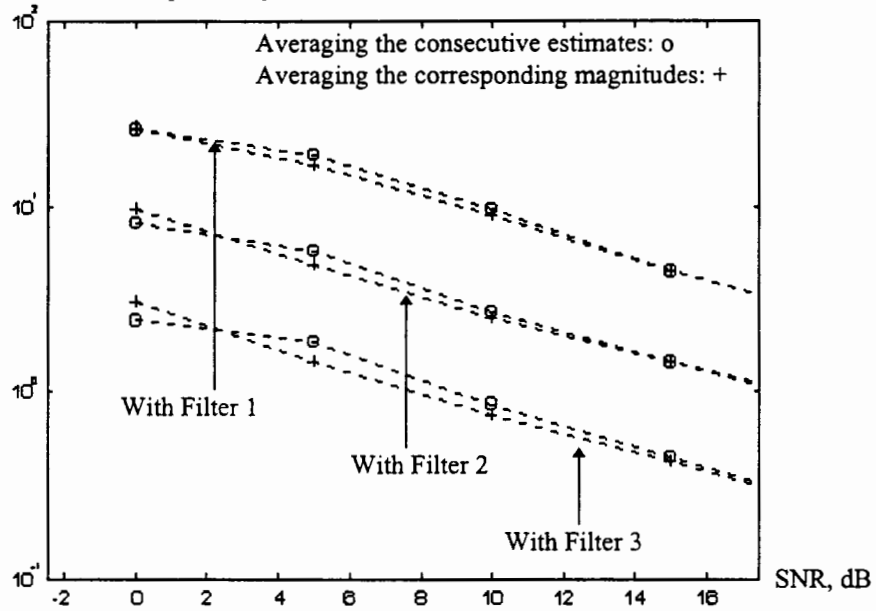

Figure 18. The performances of the Magnitude method AFC with filters by averaging the predictions and by averaging the corresponding magnitudes.

The data is shown in Table V.

TABLE V

THE PERFORMANCES OF THE MAGNITUDE METHOD AFC WITH TWO AVERAGING PROCESSES

	Steady state RMS error as the percentage of the bandwidth of the carrier band	
SNR with Filter 1	Averaging the consecutive estimates	Averaging the corresponding magnitudes
0 dB	26.0180%	26.7059%
5 dB	18.9599%	16.8088%
10 dB	9.6602%	9.0833%
15 dB	4.5462%	4.4860%

	Steady state RMS error as the percentage of the bandwidth of the carrier band	
SNR with Filter 2	Averaging the consecutive estimates	Averaging the corresponding magnitudes
0 dB	8.2009%	9.7584%
5 dB	5.8091%	4.9791%
10 dB	2.7577%	2.5390%
15 dB	1.4396%	1.4677%

	Steady state RMS error as the percentage of the bandwidth of the carrier band	
SNR with Filter 3	Averaging the consecutive estimates	Averaging the corresponding magnitudes
0 dB	2.4412%	3.0848%
5 dB	1.8582%	1.4665%
10 dB	0.8615%	0.7593%
15 dB	0.4451%	0.4318%

Finally, the performances of both AFC methods with filters are shown in Figure 19.

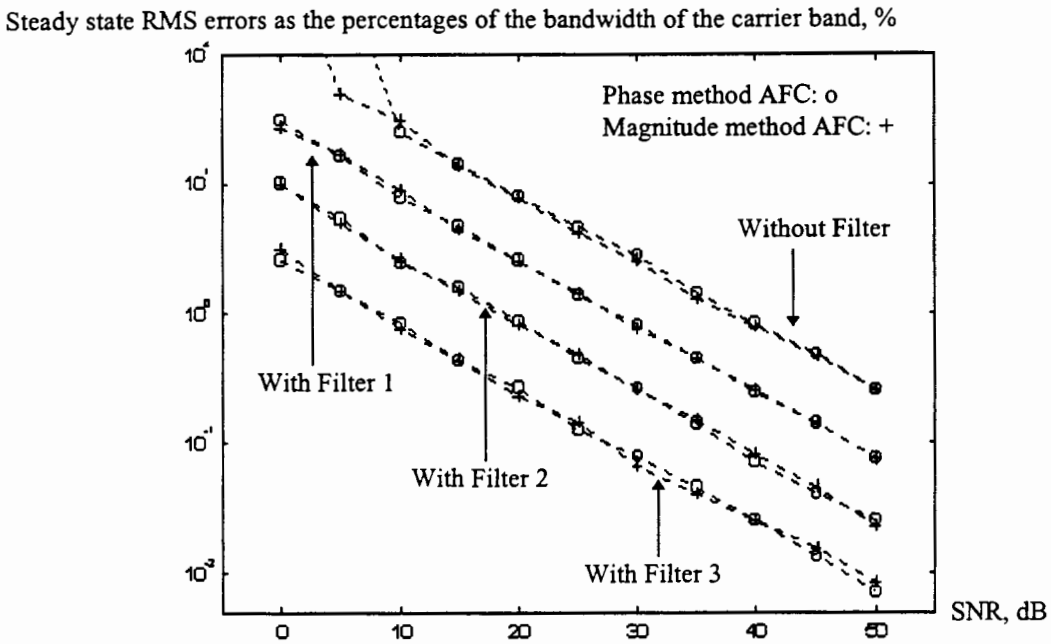

Figure 19. The comparison of the performances of both the Phase method AFC and the Magnitude method AFC with the Filter 1, 2, and 3, respectively.

Both AFC methods with filters perform equally well.

SUMMARY

The SNR can be improved by using the averaging filter effectively. The contribution in this chapter is that, for both methods, 10 dB SNR improvement can be achieved by increasing the filter length by one order of magnitude. However, in order to maintain the performances of both digital AFC methods with filters, the averaging processes have to be kept continuously. In the next chapter, the adaptive AFC algorithm will be studied and proposed to minimize the frequency errors.

CHAPTER V

ADAPTIVE AUTOMATIC FREQUENCY CONTROL ALGORITHM

INTRODUCTION

In the non-adaptive Magnitude method AFC, the local frequency is adjusted by the amount $\Delta\hat{f}$ after each new $\Delta\hat{f}$ is computed, i.e.,

$$f_L(next) = f_L(previous) - \Delta\hat{f}$$

As observed in the previous section, if the SNR is not adequate, $\Delta\hat{f}$ can fluctuate greatly from estimate to estimate due to the noise. This tends to limit the performances of the AFC loop [4].

One possible way to improve the performances without filters is to decrease the amount of the frequency adjustment by controlling the step size as follow.

$$f_L(next) = \begin{cases} f_L(previous) - \Delta\hat{f} & if \ \left|\Delta\hat{f}\right| < \alpha \\ f_L(previous) - \alpha \cdot sign\left(\Delta\hat{f}\right) & if \ \left|\Delta\hat{f}\right| \geq \alpha \end{cases}$$

where α is an adjustable step size. Normally α will be large in the beginning and become smaller when the local frequency is close to the input carrier frequency. To describe the algorithm clearly, some notations have to be defined.

NOTATIONS

The notations and parameters used in the adaptive AFC algorithm are described as follow.

f_L: the local frequency value.

f_{HIGH}: the highest frequency value of the carrier band.

f_{LOW}: the lowest frequency value of the carrier band.

Step_Margin: the allowable maximum frequency adjustment to obtain the next local frequency value.

Upper_Bound: the allowable highest frequency value to obtain the next local frequency value.

Lower_Bound: the allowable lowest frequency value to obtain the next local frequency value.

Expanding_factor: a ratio of the next *Step_Margin* to the current *Step_Margin* when the adaptive AFC determines to increase the current *Step_Margin*.

Shrinking_factor: a ratio of the next *Step_Margin* to the current *Step_Margin* when the adaptive AFC determines to decrease the current *Step_Margin*.

Hit_Up: a counter which value is increased by 1 if the next local frequency estimated by the Magnitude method AFC is equal to or higher than the current *Upper_Bound*.

Hit_Low: a counter which value is increased by 1 if the next local frequency estimated

by the Magnitude method AFC is equal to or lower than the current *Lower_Bound*.

Hit_None: a counter which value is increased by 1 if the next local frequency estimated by the Magnitude AFC is between the current *Upper_Bound* and *Lower_Bound*.

RANGES OF THE PARAMETERS

Assume that the incoming carrier frequency is stable and within the carrier band. This information can be used to obtain the ranges of the following parameters.

(1) The local frequency value is always within the carrier band.

$$f_{LOW} \leq f_L \leq f_{HIGH}$$

(2) *Step_Margin* is a positive frequency value which is less than the bandwidth of the carrier band.

$$0 < Step\_Margin \leq f_{HIGH} - f_{LOW}$$

(3) *Upper_Bound* is always lower than or equal to f_{HIGH}.

$$Upper\_Bound \leq f_{HIGH}$$

(4) *Lower_Bound* is always higher than or equal to f_{LOW}.

$$Lower\_Bound \geq f_{LOW}$$

(5) *Expanding_factor* is larger than or equal to 1.

$$Expanding\_factor \geq 1$$

(6) *Shrinking_factor* is between 0 and 1.

$$0 < Shrinking\_factor \leq 1$$

INITIAL CONDITIONS

The initial values of the parameters of the adaptive AFC algorithm are as follow.

$f_L(initial)$ = a frequency value within the carrier band

$Step\_Margin(initial)$ = the bandwidth of the carrier band = $f_{HIGH} - f_{LOW}$

$Upper\_Bound(initial) = f_{HIGH}$

$Lower\_Bound(initial) = f_{LOW}$

$Expanding\_factor(initial)$ = a value in its range

$Shrinking\_factor(initial)$ = a value in its range

$Hit\_Up(initial) = 0$

$Hit\_Low(initial) = 0$

$Hit\_None(initial) = 0$

PROGRAM

To complete one iteration of the adaptive AFC algorithm, there are five operations.

Determine the Next Local Frequency Value

When the Magnitude method AFC makes an estimate of the next local frequency, this frequency of estimate will be adjusted by the adaptive AFC according to the

following rules. If the frequency of estimate is higher than *Upper_Bound*, then the next local frequency is *Upper_Bound*. If the frequency of estimate is within *Upper_Bound* and *Lower_Bound*, then the next local frequency is the frequency of estimate. If the frequency of estimate is lower than *Lower_Bound*, then the next local frequency is *Lower_Bound*.

Adjust *Hit_Up* Counter, *Hit_Low* Counter, And *Hit_None* Counter

The rules to adjust *Hit_Up* counter, *Hit_Low* counter, and *Hit_None* counter have been described earlier, and here are some additional restrictions. If the value of any one of the three counters is increased by 1, then the values of the other two counters are set to be 0. If the value of any one of the three counters reaches its threshold value, then the values of all three counters are set to be 0 and the next *Step_Margin* will be adjusted as follow.

Adjust the Next *Step_Margin*

The following rules are used to adjust *Step_Margin*. If *Hit_Up* counter or *Hit_Low* counter reaches its threshold value, then the next *Step_Margin* is obtained by multiplying the current *Step_Margin* by *Expanding_factor*. If the *Hit_None* counter reaches its threshold value, then the next *Step_Margin* is obtained by multiplying the current *Step_Margin* by *Shrinking_factor*. If none of three counters reach its threshold value, then *Step_Margin* remains unchanged.

Move the Next *Upper_Bound*

After the next local frequency and the next *Step_Margin* are obtained, the adaptive AFC will move the next *Upper_Bound* as follow. If the value of the next local frequency plus *Step_Margin* is higher than f_{HIGH}, then the next *Upper_Bound* is f_{HIGH}. If the value of the next local frequency plus *Step_Margin* is lower than or equal to f_{HIGH}, then the next *Upper_Bound* is the next local frequency plus *Step_Margin*.

Move the Next *Lower_Bound*

After the next local frequency and *Step_Margin* are obtained, the adaptive AFC will move the next *Lower_Bound* as follow. If the value of the next local frequency minus *Step_Margin* is lower than f_{LOW}, then the next *Lower_Bound* is f_{LOW}. If the value of the next local frequency minus *Step_Margin* is higher than or equal to f_{LOW}, then the next *Lower_Bound* is the next local frequency minus *Step_Margin*.

After one iteration of the adaptive AFC algorithm is completed, the next local frequency is sent to the NCO. The flowchart for the adaptive AFC algorithm is shown in Figure 20.

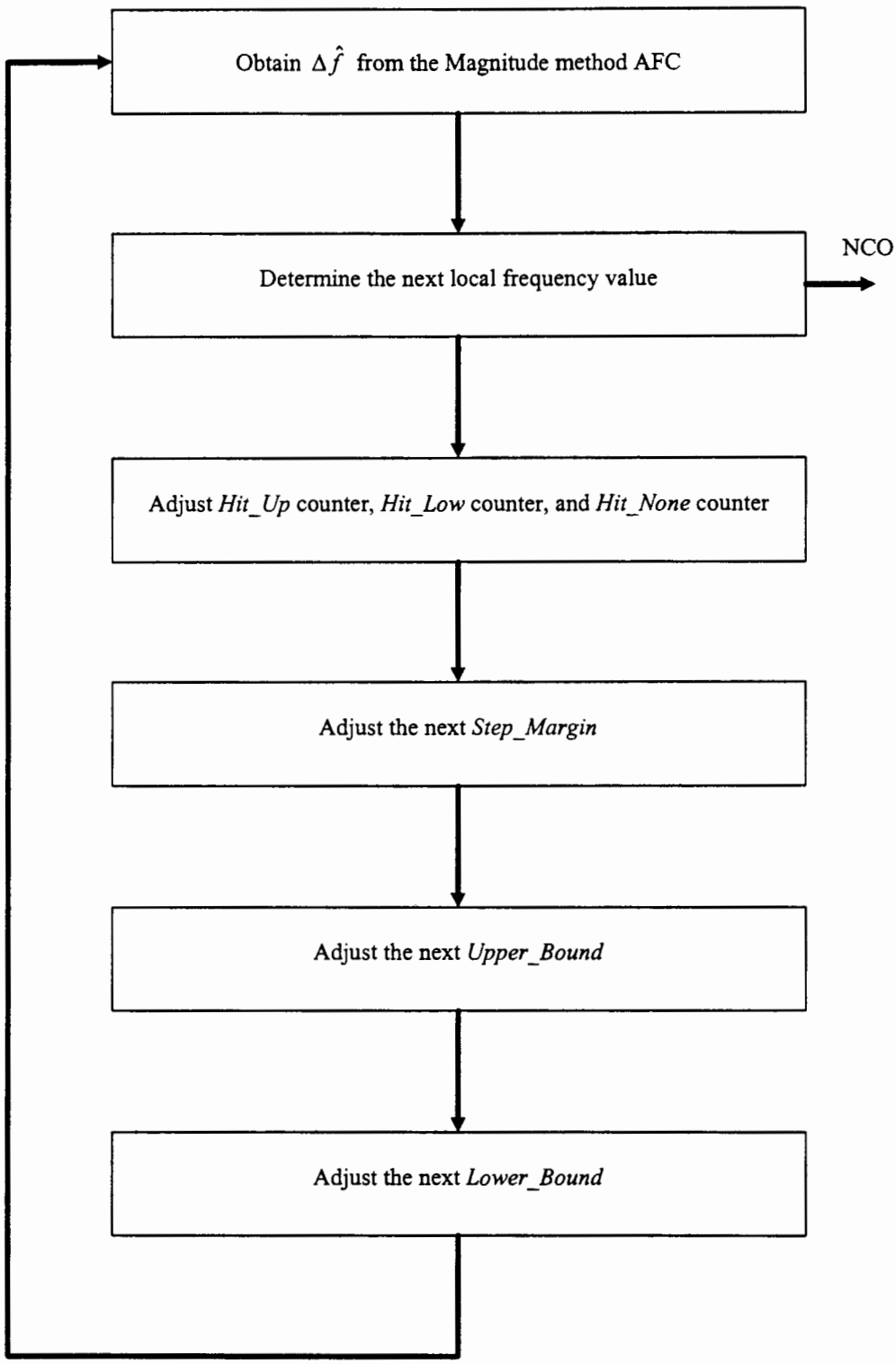

Figure 20. The flowchart for the adaptive AFC algorithm.

EFFECTS OF THE PARAMETERS

Study indicates that different performances of the adaptive AFC are affected by different combinations of the parameters, such as *Expanding_factor*, *Shrinking_factor*, *Hit_Up* counter, *Hit_Low* counter, and *Hit_None* counter. In the later simulation, three different configurations are chosen to demonstrate the effects of those parameters. The three configurations are shown in Table VI.

TABLE VI

DIFFERENT ADAPTIVE AFC CONFIGURATIONS

Adaptive AFC configuration	First	Second	Third
Expanding_factor	2	1.1	1.1
The threshold value of the *Hit_Up* counter	3	3	10
The threshold value of the *Hit_Low* counter	3	3	10
Shrinking_factor	0.9	0.9	0.9
The threshold value of the *Hit_None* counter	3	3	1

In the first configuration, the adaptive AFC will expand *Step_Margin* by multiplying the current *Step_Margin* by 2 if the next local frequency has been hitting the previous *Upper_Bound* 3 times continuously or has been hitting the previous *Lower_Bound* 3 times continuously. On the other hands, the adaptive AFC will shrink *Step_Margin* by multiplying the current *Step_Margin* by 0.9 if the next local frequency has been hitting neither the previous *Upper_Bound* nor the previous

Lower_Bound 3 times continuously. In the second configuration, all the parameters are the same as the ones in the first configuration except *Expanding_factor* which is smaller. In other words, the amount of increasing *Step_Margin* in the second configuration is smaller than the one in the first configuration. In the third configuration, the amount and the chance of increasing *Step_Margin* are much smaller than the ones in the first configuration, as indicated by the higher threshold values of *Hit_Up* counter and of *Hit_Low* counter.

Performances at the Steady State

The performances of the adaptive AFC algorithm at the steady state are simulated and measured as follow. The initial conditions of the adaptive AFC in the simulations are the same as the ones in the Magnitude method AFC. The frequency interval of the carrier band is from 97.5 MHz to 102.5 MHz. The desired carrier frequency is 100 MHz. The initial local frequency is 98 MHz. The lowest sampling rate has to be 4(102.5-97.5)=20 MHz. The sampling rate used in the simulation is 25 MHz. The performances of the adaptive AFC of different configurations are shown in Figure 21.

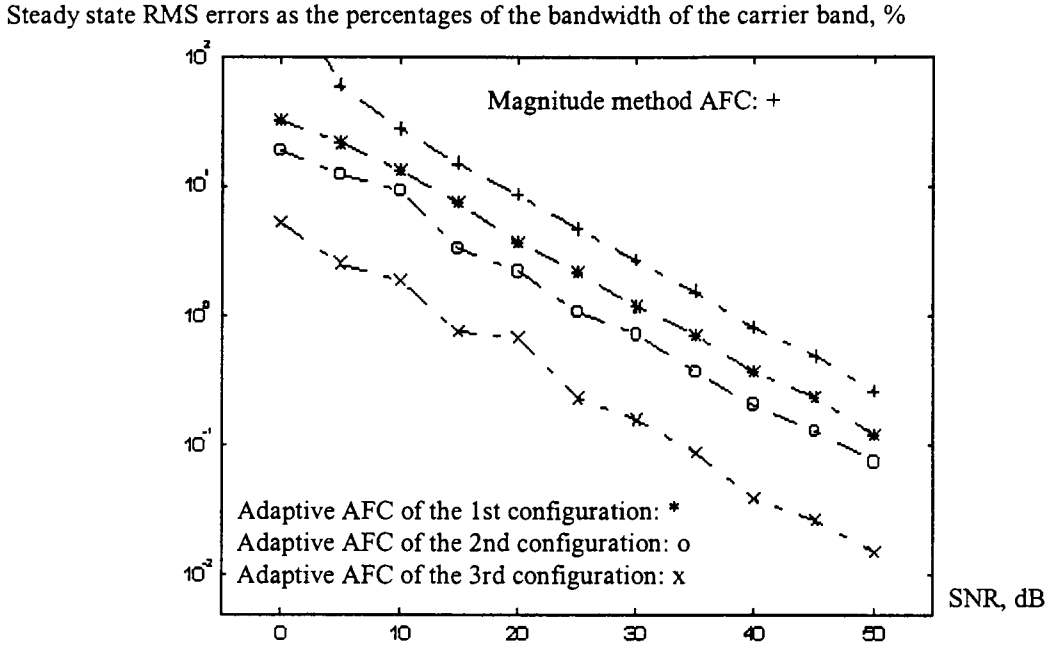

Figure 21. The performances of the adaptive AFC algorithm.

The data are shown in Table VII.

TABLE VII

THE PERFORMANCES OF THE ADAPTIVE AFC ALGORITHM

	Steady state RMS error as the percentage of the bandwidth of the carrier band		
SNR	1st configuration	2nd configuration	3rd configuration
0 dB	31.9282%	18.7645%	5.1809%
5 dB	21.3128%	12.3719%	2.5465%
10 dB	13.1789%	9.3377%	1.8851%
15 dB	7.4828%	3.3708%	0.7675%
20 dB	3.7392%	2.2153%	0.6790%
25 dB	2.1801%	1.0918%	0.2324%
30 dB	1.1793%	0.7111%	0.1568%
35 dB	0.6970%	0.3665%	0.0885%
40 dB	0.3731%	0.2083%	0.0386%
45 dB	0.2378%	0.1291%	0.0263%
50 dB	0.1196%	0.0750%	0.0153%

The adaptive AFC of the first, the second, and the third configurations can achieve, approximately, 5 dB, 10 dB, and 20 dB SNR improvements over the Magnitude method AFC, respectively. Especially, in the adaptive AFC of the third configuration, the allowable steps become smaller gradually and become larger slowly. Therefore, the adaptive AFC of the third configuration has the least RMS errors if the input carrier frequency is stable.

Performances at the Transient State

The first and the third configurations are chosen to show the performances of the adaptive AFC algorithm at the transient state. The amount and the chance of increasing *Step_Margin* in the third configuration are smaller than the ones in the first

configurations. There are more iterations for the third configuration to catch the unstable carrier frequency. Therefore, in the transient state, the third configuration needs more samples than the first configuration does if the input carrier has a step change in frequency. The initial conditions of the simulations are as follow. The frequency interval of the carrier band is from 97.5 MHz to 102.5 MHz. The desired carrier frequency is 100 MHz. The initial local frequency is 98 MHz. The lowest sampling rate has to be 4(102.5-97.5)=20 MHz. The sampling rate used in the simulation is 25 MHz. The SNR is 30 dB. After 500 samples used by the adaptive AFC algorithm, there is a step change of the carrier frequency from 100 MHz to 99 MHz. The results are shown in Figure 22.

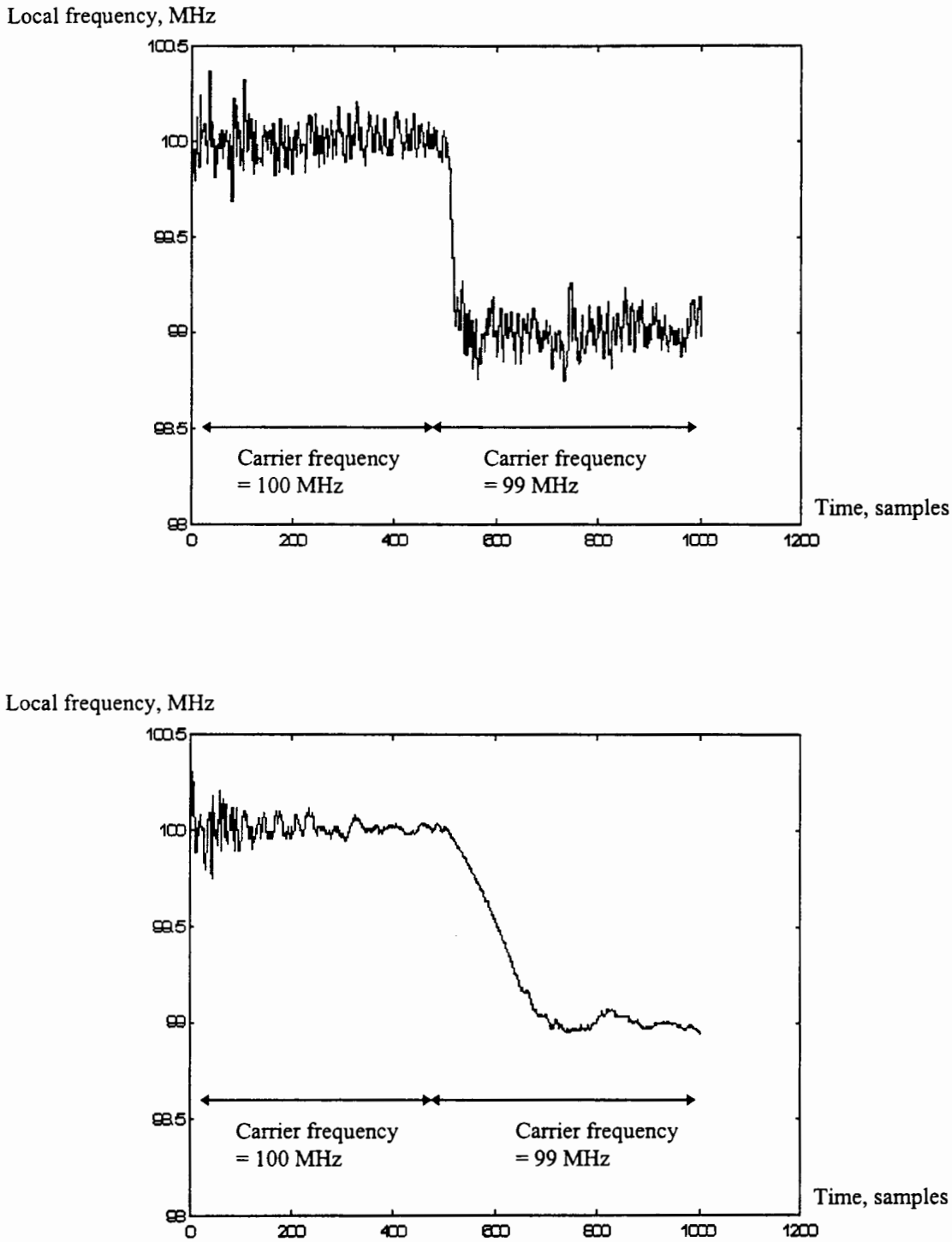

Figure 22. The performances of the adaptive AFC algorithm if there is a step change of the carrier frequency. (Top) of the first configuration. (Bottom) of the third configuration.

SUMMARY

In this chapter, the adaptive AFC is studied and the algorithm is proposed. The frequency errors are attenuated by limiting the frequency adjustments of the local oscillator. The contributions in this chapter are that the performances of the adaptive AFC algorithm are affected by the different combinations of the factors and the counters. The abilities of the adaptive AFC algorithm to catch the unstable carrier frequency are determined mostly by the factors. The performances of the adaptive AFC algorithm to hold the stable carrier frequency are determined mostly by the counters. Also, the other contribution is that the adaptive AFC algorithm with different configurations has, approximately, 5 to 20 dB SNR improvement over the Magnitude method AFC.

CHAPTER VI

CONCLUSIONS

Aliasing sampling is an attractive way to demodulate the narrowband radio frequency signals with high frequency carriers. Since the uncertainty of the oscillator drifting is not avoidable, error will occur. The sensitivity of the aliasing sampling to the frequency error is investigated. Under the assumption of the one known baseband signal, the ESR is predicted at different sampling rates and at different carrier frequencies.

To compensate the frequency error, two digital AFC algorithms are studied. The performances of the Magnitude method AFC are better than the ones of the Phase method AFC. Although both digital AFC method can catch the carrier frequency by using two samples, they are sensitive to the noise, especially, at low SNR. It has been shown that the averaging filters can improve the noise immunity according to the length of the averaging filters. Specifically, a 10 dB SNR improvement can be achieved by increasing the filter length by one order of magnitude.

Then the adaptive AFC is studied and one algorithm is proposed. It is shown that the adaptive AFC algorithm has, approximately, 5 to 20 dB SNR improvement over the Magnitude method AFC with different parameter configurations. The performances of the adaptive AFC algorithm are affected by the different

combinations of the factors and counters. The parameters of the adaptive AFC algorithm of each configuration are set to be constants at all time. Further research will be to investigate the fast adaptive AFC by automatically varying those parameters according to its operating environment.

REFERENCES

[1] Ferrel G. Stremler. Introduction to Communication Systems, 3rd Ed. Addison-Wesley Publishing Co. 1990.

[2] M. J. Underhill. Fundamentals of oscillator performance. Electronics & Communication Engineering Journal, 185-193. August 1992.

[3] P. M. Grant. Multirate signal processing. Electronics & Communication Engineering Journal, 4-12. February 1996.

[4] Floyd M. Gardner. Properties of frequency difference detectors. IEEE Transactions on Communications, Volume COM-33, Number 2, 131-138, February 1985.

[5] Ulrich L. Rohde and T. T. N. Bucher. Communications Receivers Principles and Design. McGraw-Hill, Inc. New York, NY 1988.

[6] Francis D. Natali. AFC Tracking Algorithms. IEEE Transactions on Communications, Volume COM-33, Number 8, 935-947, August 1984.

[7] John B. Thomas. An introduction to Statistical Communication Theory. John Wiley & Son, Inc. 1969.

APPENDIX

The following main programs are used to generate data for the ESR, the Phase method AFC, the Magnitude method AFC, the averaging filters, and the adaptive AFC algorithm.

ESR FROM 2 Hz TO 20 Hz

```
clf
clear
fl=2;
fr=20;
fdiv=.01;
tl=-5;
tr=5;
for fs=fl:fdiv:fr
        t=1/fs*round(tl*fs): 1/fs: 1/fs*round(tr*fs);
        f=sin(pi*t).*sin(pi*t)/pi/pi./t./t;
        f(round(5*fs)+1)=1;
        fm=f.*cos(2*pi*20*t);
        ntos((fs-fl)/fdiv +1)=((f-fm)*(f-fm)')/(f*f');
end
freq=fl:fdiv:fr;
plot(freq,ntos*100, 'w-')
axis([2 20 0 200])
set(1, 'PaperPosition', [0 0 4.8 3.6])
```

ESR UNDER THE SAME CARRIER FREQUENCY

```
clf
clear
subplot(211)
fl=9.5;
fr=10.5;
fdiv=.01;
for fs=fl:fdiv:fr
        t=-1/fs*round(5*fs): 1/fs: 1/fs*round(5*fs);
        f=sin(pi*t).*sin(pi*t)/pi/pi./t./t;
        f(round(5*fs)+1)=1;
        fm=f.*cos(2*pi*20*t);
        ntos((fs-fl)/fdiv +1)=((f-fm)*(f-fm)')/(f*f');
end
freq=fl:fdiv:fr;
plot(freq,ntos*100, 'w-')

subplot(212)
fl=3.8;
fr=4.2;
fdiv=.004;
for fs=fl:fdiv:fr
        t=-1/fs*round(5*fs): 1/fs: 1/fs*round(5*fs);
        f=sin(pi*t).*sin(pi*t)/pi/pi./t./t;
        f(round(5*fs)+1)=1;
        fm=f.*cos(2*pi*20*t);
        ntos1((fs-fl)/fdiv +1)=((f-fm)*(f-fm)')/(f*f');
end
freq=fl:fdiv:fr;
plot(freq,ntos1*100, 'w-')
axis([3.8 4.2 0 100])

set(1, 'PaperPosition', [0 0 4.8 3.6])
```

ESR UNDER THE DESIRED ALIASING SAMPLING RATE

```
clf
clear
subplot(211)
fl=9.95;
fr=10.05;
fdiv=.001;
for fs=fl:fdiv:fr
        t=-1/fs*round(5*fs): 1/fs: 1/fs*round(5*fs);
        f=sin(pi*t).*sin(pi*t)/pi/pi./t./t;
        f(round(5*fs)+1)=1;
        fm=f.*cos(2*pi*200*t);
        ntos1((fs-fl)/fdiv +1)=((f-fm)*(f-fm)')/(f*f');
end
freq=fl:fdiv:fr;
plot(freq,ntos1*100, 'w-')
axis([9.95 10.05 0 100])

subplot(212)
fl=9.995;
fr=10.005;
fdiv=.0001;
for fs=fl:fdiv:fr
        t=-1/fs*round(5*fs): 1/fs: 1/fs*round(5*fs);
        f=sin(pi*t).*sin(pi*t)/pi/pi./t./t;
        f(round(5*fs)+1)=1;
        fm=f.*cos(2*pi*2000*t);
        ntos2((fs-fl)/fdiv +1)=((f-fm)*(f-fm)')/(f*f');
end
freq=fl:fdiv:fr;
plot(freq,ntos2*100, 'w-')
axis([9.995 10.005 0 100])

set(1, 'PaperPosition', [0 0 4.8 3.6])
```

ESR AS THE FUNCTION OF THE FREQUENCY DEVIATION PERCENTAGE

```
clear
fl=9.5;
fr=10.5;
fdiv=.005;
for fs=fl:fdiv:fr
        t=-1/fs*round(5*fs): 1/fs: 1/fs*round(5*fs);
        f=sin(pi*t).*sin(pi*t)/pi/pi./t./t;
        f(round(5*fs)+1)=1;
        fm=f.*cos(2*pi*20*t);
        ntos((fs-fl)/fdiv +1)=((f-fm)*(f-fm)')/(f*f');
end
clf
fdev=-100:.05*20:100;
ntos1=ntos*100;
plot(fdev,ntos1, 'w-')
set(1, 'PaperPosition', [0 0 4.8 3.6])
```

PHASE METHOD AFC ALGORITHM

```
clear
for db=0:5:50,
db
fs=25*10^6;
fc=100*10^6;
fl(1)=98*10^6;
for n=0:2:1600,

 theta=rand*2*pi;
 phi1=rand*2*pi;
 phi2=rand*2*pi;
 noisea=randn(1,2);
 noiseb=randn(1,2);
 noisea=noisea*sqrt(1/(10^(db/10))/2);
 noiseb=noiseb*sqrt(1/(10^(db/10))/2);
 noise=sqrt(noisea.*noisea + noiseb.*noiseb);

 fl(n+2)=fl(n+1);
 x(1)=exp(j*(2*pi*(fl(n+1) - fc)*n/fs + theta)) + noise(1)*exp(j*(2*pi*(fl(n+1) -
fc)*n/fs + phi1));
 x(2)=exp(j*(2*pi*(fl(n+2) - fc)*(n+1)/fs + theta)) + noise(2)*exp(j*(2*pi*(fl(n+2) -
fc)*(n+1)/fs + phi2));
 deltaphase=angle(x(2)) - angle(x(1));
 if deltaphase > pi
  deltaphase=deltaphase-2*pi;
 end
 if deltaphase < -pi
  deltaphase=deltaphase+2*pi;
 end
 fl(n+3)=fl(n+2) - fs*deltaphase/2/pi;
end

PFNmse(db/5+1)=1/length(fl(1000:1500))*((fl(1000:1500)-fc)*(fl(1000:1500)-fc)');
plot(fl)
pause
clear fl

%
end
save PFNmse PFNmse
```

```
db=0:5:50;
figure(1)
semilogy(db, sqrt(PFNmse)/10^6, 'wo')
hold on
semilogy(db, sqrt(PFNmse)/10^6, 'w:')
axis([-5 55 .001 5])
set(1, 'PaperPosition', [0 0 4.8 3.6])
%
clear db
clear deltaphase
clear fc
clear fl
clear fs
clear n
clear noise
clear noisea
clear noiseb
clear phi1
clear phi2
clear theta
clear x
```

PERFORMANCE OF THE PHASE METHOD AFC WITH NOISE

```
clear
clf
figure(1)
load PFNmse
fb=5*10^6;
db=0:5:50;
semilogy(db, sqrt(PFNmse)/fb*100, 'ow')
hold on
semilogy(db, sqrt(PFNmse)/fb*100, ':w')
axis([-5 55 .1 100])
set(1, 'PaperPosition', [0 0 4.8 3.6])
```

DTFT OF THE MAGNITUDE METHOD AFC WITH AND WITHOUT NOISE

```
clear
clf
subplot(221)
fs=25*10^6;
fc=100*10^6;
fl(1)=104*10^6;
n=0;
 theta=rand*2*pi;
 fl(n+2)=fl(n+1);
 x(1)=exp(j*(2*pi*(fl(n+1) - fc)*n/fs + theta));
 x(2)=exp(j*(2*pi*(fl(n+2) - fc)*(n+1)/fs + theta));
 x(3)=0;
 x(4)=0;
 X=abs(fft(x));
 xstem=[0 .25 .5 .75];
 stem(xstem, X)
 hold on
 plot(xstem, X, 'wo')
 xone(1)=x(1);
 xone(2)=x(2);
 for m=3:100,
  xone(m)=0;
 end
 Xone=abs(fft(xone));
 All(1:100)=Xone;
 All(101:200)=Xone;
 xAll=-1:1/100:1-1/100;
 plot(xAll,All, 'w-')
 axis([-1 1 0 3])

subplot(223)
plot(xAll, All, 'w:')
axis([-1 1 0 3])
hold on
clear
db=5;
fs=25*10^6;
fc=100*10^6;
fl(1)=104*10^6;
n=0;
```

```
  theta=rand*2*pi;
  phi1=rand*2*pi;
  phi2=rand*2*pi;
  noisea=randn(1,2);
  noiseb=randn(1,2);
  noisea=noisea*sqrt(1/(10^(db/10))/2);
  noiseb=noiseb*sqrt(1/(10^(db/10))/2);
  noise=sqrt(noisea.*noisea + noiseb.*noiseb);

  fl(n+2)=fl(n+1);
  x(1)=exp(j*(2*pi*(fl(n+1) - fc)*n/fs + theta)) + noise(1)*exp(j*(2*pi*(fl(n+1) -
fc)*n/fs + phi1));
  x(2)=exp(j*(2*pi*(fl(n+2) - fc)*(n+1)/fs + theta)) + noise(2)*exp(j*(2*pi*(fl(n+2) -
fc)*(n+1)/fs + phi2));
  x(3)=0;
  x(4)=0;
  X=abs(fft(x));
  xstem=[0 .25 .5 .75];
  stem(xstem, X)
  plot(xstem, X, 'wo')
  xone(1)=x(1);
  xone(2)=x(2);
  for m=3:100,
    xone(m)=0;
  end
  Xone=abs(fft(xone));
  All(1:100)=Xone;
  All(101:200)=Xone;
  xAll=-1:1/100:1-1/100;
  plot(xAll,All, 'w-')
set(1, 'PaperPosition', [0 0 4.8 3.6])
figure(1)
```

MAGNITUDE METHOD AFC ALGORITHM

```
clear
for db=0:5:50,
db
fs=25*10^6;
fc=100*10^6;
fl(1)=98*10^6;
for n=0:2:1550,

 theta=rand*2*pi;
 phi1=rand*2*pi;
 phi2=rand*2*pi;
 noisea=randn(1,2);
 noiseb=randn(1,2);
 noisea=noisea*sqrt(1/(10^(db/10))/2);
 noiseb=noiseb*sqrt(1/(10^(db/10))/2);
 noise=sqrt(noisea.*noisea + noiseb.*noiseb);

 fl(n+2)=fl(n+1);
 x(1)=exp(j*(2*pi*(fl(n+1) - fc)*n/fs + theta)) + noise(1)*exp(j*(2*pi*(fl(n+1) -
fc)*n/fs + phi1));
 x(2)=exp(j*(2*pi*(fl(n+2) - fc)*(n+1)/fs + theta)) + noise(2)*exp(j*(2*pi*(fl(n+2) -
fc)*(n+1)/fs + phi2));
 x(3)=0;
 x(4)=0;
 X=abs(fft(x));

 deltafTs = 1/2/pi*real(asin(X(2)*X(2)/4 - X(4)*X(4)/4));

 fl(n+3)=fl(n+2) - fs*deltafTs;

end  %of n

JFNmse(db/5+1)=1/length(fl(1000:1500))*((fl(1000:1500)-fc)*(fl(1000:1500)-fc)');
figure(2)
plot(fl/10^6)
pause
clear fl
end  %of db

save JFNmse JFNmse
```

```
db=0:5:50;
figure
semilogy(db, sqrt(JFNmse)/fc*100, '+')
hold on
semilogy(db, sqrt(JFNmse)/fc*100, ':')
%
clear db
clear deltafTs
clear fc
clear fl
clear fs
clear n
clear noise
clear noisea
clear noiseb
clear phi1
clear phi2
clear theta
clear x
clear X
who


%
```

```
clear
clf
figure(1)
load JFNmse
fb=5*10^6;
db=0:5:50;
semilogy(db, sqrt(JFNmse)/fb*100, '+w')
hold on
semilogy(db, sqrt(JFNmse)/fb*100, ':w')
axis([-5 55 .1 100])
set(1, 'PaperPosition', [0 0 4.8 3.6])
```

PHASE METHOD AFC WITH FILTER 1 ALGORITHM

```
!date
clear
who
for db=0:5:50,
db
fs=25*10^6;
fc=100*10^6;
fl(1)=98*10^6;

for n=0:20:11020,

  for o=0:2:18,
    theta=rand*2*pi;
    phi1=rand*2*pi;
    phi2=rand*2*pi;
    noisea=randn(1,2);
    noiseb=randn(1,2);
    noisea=noisea*sqrt(1/(10^(db/10))/2);
    noiseb=noiseb*sqrt(1/(10^(db/10))/2);
    noise=sqrt(noisea.*noisea + noiseb.*noiseb);

    x(1)=exp(j*(2*pi*(fl(n+o+1) - fc)*(n+o)/fs + theta)) +
noise(1)*exp(j*(2*pi*(fl(n+o+1) - fc)*(n+o)/fs + phi1));
    fl(n+o+2)=fl(n+o+1);

    x(2)=exp(j*(2*pi*(fl(n+o+2) - fc)*(n+o+1)/fs + theta)) +
noise(2)*exp(j*(2*pi*(fl(n+o+2) - fc)*(n+o+1)/fs + phi2));
    fl(n+o+3)=fl(n+o+2);

    deltaphase(o/2+1)=angle(x(2)) - angle(x(1));
    if deltaphase(o/2+1) >pi
      deltaphase(o/2+1)=deltaphase(o/2+1)-2*pi;
    end
    if deltaphase(o/2+1) <-pi
      deltaphase(o/2+1)=deltaphase(o/2+1)+2*pi;
    end

  end  %end of o

  fl(n+21)=fl(n+20) - fs*mean(deltaphase)/2/pi;
```

```
 clear deltaphase
end  %end of n

PF1mse(db/5+1)=1/length(fl(1001:11000))*((fl(1001:11000)-fc)*(fl(1001:11000)-
fc)');

save PF1mse PF1mse
plot(fl)
pause
clear fl

end  %end of db

db=0:5:50;

semilogy(db, sqrt(PF1mse)/fc*100, 'o')
hold on
semilogy(db, sqrt(PF1mse)/fc*100, ':')

clear noisea
clear phi2
clear db
clear fs
clear noiseb
clear theta
clear deltaphase
clear n
clear o
clear x
clear fc
clear noise
clear phi1
!date
%
```

PHASE METHOD AFC WITH FILTER 2 ALGORITHM

```
!date
clear
who
for db=0:5:50,
db
fs=25*10^6;
fc=100*10^6;
fl(1)=98*10^6;

for n=0:200:22200,
n
  for o=0:2:198,
    theta=rand*2*pi;
    phi1=rand*2*pi;
    phi2=rand*2*pi;
    noisea=randn(1,2);
    noiseb=randn(1,2);
    noisea=noisea*sqrt(1/(10^(db/10))/2);
    noiseb=noiseb*sqrt(1/(10^(db/10))/2);
    noise=sqrt(noisea.*noisea + noiseb.*noiseb);

    x(1)=exp(j*(2*pi*(fl(n/200+1) - fc)*(n+o)/fs + theta)) +
noise(1)*exp(j*(2*pi*(fl(n/200+1) - fc)*(n+o)/fs + phi1));

    x(2)=exp(j*(2*pi*(fl(n/200+1) - fc)*(n+o+1)/fs + theta)) +
noise(2)*exp(j*(2*pi*(fl(n/200+1) - fc)*(n+o+1)/fs + phi2));

    deltaphase(o/2+1)=angle(x(2)) - angle(x(1));
    if deltaphase(o/2+1) >pi
      deltaphase(o/2+1)=deltaphase(o/2+1)-2*pi;
    end
    if deltaphase(o/2+1) <-pi
      deltaphase(o/2+1)=deltaphase(o/2+1)+2*pi;
    end
  end  %end of o

  fl(n/200+2)=fl(n/200+1) - fs*mean(deltaphase)/2/pi;
  clear deltaphase
end  %end of n
```

```
plot(fl)
pause
PF2mse(db/5+1)=1/length(fl(10:109))*((fl(10:109)-fc)*(fl(10:109)-fc)');

save PF2mse PF2mse
clear fl
end  %end of db

db=0:5:50;

semilogy(db, sqrt(PF2mse)/fc*100, '*')
hold on
semilogy(db, sqrt(PF2mse)/fc*100, ':')


clear noisea
clear phi2
clear db
clear fs
clear noiseb
clear theta
clear deltaphase
clear n
clear o
clear x
clear fc
clear noise
clear phi1
!date
%
```

PHASE METHOD AFC WITH FILTER 3 ALGORITHM

```
!date
clear
who
for db=0:5:50,
db
fs=25*10^6;
fc=100*10^6;
fl(1)=98*10^6;

for n=0:2000:222000,
n
 for o=0:2:1998,
  theta=rand*2*pi;
  phi1=rand*2*pi;
  phi2=rand*2*pi;
  noisea=randn(1,2);
  noiseb=randn(1,2);
  noisea=noisea*sqrt(1/(10^(db/10))/2);
  noiseb=noiseb*sqrt(1/(10^(db/10))/2);
  noise=sqrt(noisea.*noisea + noiseb.*noiseb);

  x(1)=exp(j*(2*pi*(fl(n/2000+1) - fc)*(n+o)/fs + theta)) +
noise(1)*exp(j*(2*pi*(fl(n/2000+1) - fc)*(n+o)/fs + phi1));

  x(2)=exp(j*(2*pi*(fl(n/2000+1) - fc)*(n+o+1)/fs + theta)) +
noise(2)*exp(j*(2*pi*(fl(n/2000+1) - fc)*(n+o+1)/fs + phi2));

  deltaphase(o/2+1)=angle(x(2)) - angle(x(1));
  if deltaphase(o/2+1) >pi
   deltaphase(o/2+1)=deltaphase(o/2+1)-2*pi;
  end
  if deltaphase(o/2+1) <-pi
   deltaphase(o/2+1)=deltaphase(o/2+1)+2*pi;
  end

 end  %end of o

 fl(n/2000+2)=fl(n/2000+1) - fs*mean(deltaphase)/2/pi;
 clear deltaphase
```

```
end  %end of n


PF3mse(db/5+1)=1/length(fl(10:109))*((fl(10:109)-fc)*(fl(10:109)-fc)');


save PF3mse PF3mse
plot(fl)
pause
clear fl
end  %end of db

db=0:5:50;

semilogy(db, sqrt(PF3mse)/fc*100, 'x')
hold on
semilogy(db, sqrt(PF3mse)/fc*100, ':')


clear noisea
clear phi2
clear db
clear fs
clear noiseb
clear theta
clear deltaphase
clear n
clear o
clear x
clear fc
clear noise
clear phi1
!date
%
```

MAGNITUDE METHOD AFC WITH FILTER 1 ALGORITHM

(AVERAGING ESTIMATE)

```
!date
who
for db=0:5:50;
db
fs=25*10^6;
fc=100*10^6;
fl(1)=95*10^6;

for n=0:20:11020,

  for o=0:2:18,
    theta=rand*2*pi;
    phi1=rand*2*pi;
    phi2=rand*2*pi;
    noisea=randn(1,2);
    noiseb=randn(1,2);
    noisea=noisea*sqrt(1/(10^(db/10))/2);
    noiseb=noiseb*sqrt(1/(10^(db/10))/2);
    noise=sqrt(noisea.*noisea + noiseb.*noiseb);

    x(1)=exp(j*(2*pi*(fl(n+o+1) - fc)*(n+o)/fs + theta)) +
noise(1)*exp(j*(2*pi*(fl(n+o+1) - fc)*(n+o)/fs + phi1));
    fl(n+o+2)=fl(n+o+1);

    x(2)=exp(j*(2*pi*(fl(n+o+2) - fc)*(n+o+1)/fs + theta)) +
noise(2)*exp(j*(2*pi*(fl(n+o+2) - fc)*(n+o+1)/fs + phi2));
    fl(n+o+3)=fl(n+o+2);
    x(3)=0;
    x(4)=0;
    X=abs(fft(x));

    deltafTsAve(o/2+1) = 1/2/pi*real(asin(X(2)*X(2)/4 - X(4)*X(4)/4));

  end  %of o

  fl(n+21)=fl(n+20) - fs*mean(deltafTsAve);
  clear deltafTsAve;
end  %of n
```

```
MF1mse(db/5+1)=1/length(fl(1001:11000))*((fl(1001:11000)-fc)*(fl(1001:11000)-
fc)');

save MF1mse MF1mse
plot(fl)
pause
clear fl

end  %end of db

db=0:5:50;

semilogy(db, sqrt(MF1mse)/fc*100, 'o')
hold on
semilogy(db, sqrt(MF1mse)/fc*100, ':')

clear noisea
clear phi2
clear db
clear fs
clear noiseb
clear theta
clear deltafTs
clear n
clear o
clear x
clear fc
clear noise
clear phi1
clear X
!date
```

MAGNITUDE METHOD AFC WITH FILTER 2 ALGORITHM

(AVERAGING ESTIMATE)

```
!date
who
for db=0:5:50;
db
fs=25*10^6;
fc=100*10^6;
fl(1)=95*10^6;

for n=0:200:22200,

  for o=0:2:198,
    theta=rand*2*pi;
    phi1=rand*2*pi;
    phi2=rand*2*pi;
    noisea=randn(1,2);
    noiseb=randn(1,2);
    noisea=noisea*sqrt(1/(10^(db/10))/2);
    noiseb=noiseb*sqrt(1/(10^(db/10))/2);
    noise=sqrt(noisea.*noisea + noiseb.*noiseb);

    x(1)=exp(j*(2*pi*(fl(n+o+1) - fc)*(n+o)/fs + theta)) +
noise(1)*exp(j*(2*pi*(fl(n+o+1) - fc)*(n+o)/fs + phi1));
    fl(n+o+2)=fl(n+o+1);

    x(2)=exp(j*(2*pi*(fl(n+o+2) - fc)*(n+o+1)/fs + theta)) +
noise(2)*exp(j*(2*pi*(fl(n+o+2) - fc)*(n+o+1)/fs + phi2));
    fl(n+o+3)=fl(n+o+2);
    x(3)=0;
    x(4)=0;
    X=abs(fft(x));

    deltafTsAve(o/2+1) = 1/2/pi*real(asin(X(2)*X(2)/4 - X(4)*X(4)/4));

  end  %of o

  fl(n+201)=fl(n+200) - fs*mean(deltafTsAve);
  clear deltafTsAve;
end  %of n
```

```
MF2mse(db/5+1)=1/length(fl(2001:22000))*((fl(2001:22000)-fc)*(fl(2001:22000)-
fc)');
plot(fl)
pause
save MF2mse MF2mse
clear fl

end  %end of db

db=0:5:50;

semilogy(db, sqrt(MF2mse)/fc*100, '*')
hold on
semilogy(db, sqrt(MF2mse)/fc*100, ':')

clear noisea
clear phi2
clear db
clear fs
clear noiseb
clear theta
clear deltafTs
clear n
clear o
clear x
clear fc
clear noise
clear phi1
clear X
!date

%
```

MAGNITUDE METHOD AFC WITH FILTER 3 ALGORITHM

(AVERAGING ESTIMATE)

```
!date
who
for db=0:5:50,
db
fs=25*10^6;
fc=100*10^6;
fl(1)=95*10^6;

for n=0:2000:222000,
n
 for o=0:2:1998,
   theta=rand*2*pi;
   phi1=rand*2*pi;
   phi2=rand*2*pi;
   noisea=randn(1,2);
   noiseb=randn(1,2);
   noisea=noisea*sqrt(1/(10^(db/10))/2);
   noiseb=noiseb*sqrt(1/(10^(db/10))/2);
   noise=sqrt(noisea.*noisea + noiseb.*noiseb);

   x(1)=exp(j*(2*pi*(fl(n/2000+1) - fc)*(n+o)/fs + theta)) +
noise(1)*exp(j*(2*pi*(fl(n/2000+1) - fc)*(n+o)/fs + phi1));

   x(2)=exp(j*(2*pi*(fl(n/2000+1) - fc)*(n+o+1)/fs + theta)) +
noise(2)*exp(j*(2*pi*(fl(n/2000+1) - fc)*(n+o+1)/fs + phi2));

   x(3)=0;
   x(4)=0;
   X=abs(fft(x));

   deltafTsAve(o/2+1) = 1/2/pi*real(asin(X(2)*X(2)/4 - X(4)*X(4)/4));

 end  %of o

 fl(n/2000+2)=fl(n/2000+1) - fs*mean(deltafTsAve);
 clear deltafTsAve;
end  %of n
```

```
plot(fl)
pause
MF3mse(db/5+1)=1/length(fl(10:109))*((fl(10:109)-fc)*(fl(10:109)-fc)');


save MF3mse MF3mse

clear fl
clear noisea
clear phi2
clear db
clear fs
clear noiseb
clear theta
clear deltafTs
clear n
clear o
clear x
clear fc
clear noise
clear phi1
clear X

end  %end of db

db=0:5:50;
fc=100*10^6;
semilogy(db, sqrt(MF3mse)/fc*100, 'x')
hold on
semilogy(db, sqrt(MF3mse)/fc*100, ':')

!date
```

MAGNITUDE METHOD AFC WITH FILTER 1 ALGORITHM

(AVERAGING MAGNITUDE)

```
!date
clear
who
for db=0:5:50;
db
fs=25*10^6;
fc=100*10^6;
fl(1)=98*10^6;

for n=0:20:11020,

  for o=0:2:18,
    theta=rand*2*pi;
    phi1=rand*2*pi;
    phi2=rand*2*pi;
    noisea=randn(1,2);
    noiseb=randn(1,2);
    noisea=noisea*sqrt(1/(10^(db/10))/2);
    noiseb=noiseb*sqrt(1/(10^(db/10))/2);
    noise=sqrt(noisea.*noisea + noiseb.*noiseb);

    x(1)=exp(j*(2*pi*(fl(n+o+1) - fc)*(n+o)/fs + theta)) +
noise(1)*exp(j*(2*pi*(fl(n+o+1) - fc)*(n+o)/fs + phi1));
    fl(n+o+2)=fl(n+o+1);

    x(2)=exp(j*(2*pi*(fl(n+o+2) - fc)*(n+o+1)/fs + theta)) +
noise(2)*exp(j*(2*pi*(fl(n+o+2) - fc)*(n+o+1)/fs + phi2));
    fl(n+o+3)=fl(n+o+2);
    x(3)=0;
    x(4)=0;
    X=abs(fft(x));

    X24error(o/2+1)=X(2)*X(2)/4-X(4)*X(4)/4;
%    deltafTsAve(o/2+1) = 1/2/pi*real(asin(X(2)*X(2)/4 - X(4)*X(4)/4));
    clear x
    clear X
end  %of o
```

```
 fl(n+21)=fl(n+20) - fs* 1/2/pi*real( asin(mean(X24error) ) );
 clear deltafTsAve;
 clear X24error
end  %of n

JF1mse(db/5+1)=1/length(fl(1001:11000))*((fl(1001:11000)-fc)*(fl(1001:11000)-
fc)');

save JF1mse JF1mse
plot(fl)
pause
clear fl

end  %end of db

db=0:5:50;

semilogy(db, sqrt(JF1mse)/fc*100, 'o')
hold on
semilogy(db, sqrt(JF1mse)/fc*100, ':')

clear noisea
clear phi2
clear db
clear fs
clear noiseb
clear theta
clear deltafTs
clear n
clear o
clear x
clear fc
clear noise
clear phi1
clear X
!date

%
```

MAGNITUDE METHOD AFC WITH FILTER 2 ALGORITHM

(AVERAGING MAGNITUDE)

```
!date
clear
clf
who
for db=0:5:50,
db
fs=25*10^6;
fc=100*10^6;
fl(1)=98*10^6;

for n=0:200:22200,
n
 for o=0:2:198,
   theta=rand*2*pi;
   phi1=rand*2*pi;
   phi2=rand*2*pi;
   noisea=randn(1,2);
   noiseb=randn(1,2);
   noisea=noisea*sqrt(1/(10^(db/10))/2);
   noiseb=noiseb*sqrt(1/(10^(db/10))/2);
   noise=sqrt(noisea.*noisea + noiseb.*noiseb);

   x(1)=exp(j*(2*pi*(fl(n/200+1) - fc)*(n+o)/fs + theta)) +
noise(1)*exp(j*(2*pi*(fl(n/200+1) - fc)*(n+o)/fs + phi1));

   x(2)=exp(j*(2*pi*(fl(n/200+1) - fc)*(n+o+1)/fs + theta)) +
noise(2)*exp(j*(2*pi*(fl(n/200+1) - fc)*(n+o+1)/fs + phi2));

   x(3)=0;
   x(4)=0;
   X=abs(fft(x));

   X24error(o/2+1)=X(2)*X(2)/4-X(4)*X(4)/4;
%   deltafTsAve(o/2+1) = 1/2/pi*real(asin(X(2)*X(2)/4 - X(4)*X(4)/4));
   clear x
   clear X
 end  %of o
```

```
 fl(n/200+2)=fl(n/200+1) - fs* 1/2/pi*real( asin(mean(X24error) ) );
 clear deltafTsAve;
 clear X24error
end  %of n

plot(fl/10^6)
pause
JF2mse(db/5+1)=1/length(fl(10:109))*((fl(10:109)-fc)*(fl(10:109)-fc)');

clear fl
clear noisea
clear phi2
clear db
clear fs
clear noiseb
clear theta
clear deltafTs
clear n
clear o
clear x
clear fc
clear noise
clear phi1
clear X

end  %end of db

save JF2mse JF2mse

db=0:5:50;
fc=100*10^6;
semilogy(db, sqrt(JF2mse)/fc*100, 'x')
hold on
semilogy(db, sqrt(JF2mse)/fc*100, ':')

!date

%
```

MAGNITUDE METHOD AFC WITH FILTER 3 ALGORITHM

(AVERAGING MAGNITUDE)

```
!date
clear
who
for db=0:5:50,
db
fs=25*10^6;
fc=100*10^6;
fl(1)=98*10^6;

for n=0:2000:222000,
n
 for o=0:2:1998,
   theta=rand*2*pi;
   phi1=rand*2*pi;
   phi2=rand*2*pi;
   noisea=randn(1,2);
   noiseb=randn(1,2);
   noisea=noisea*sqrt(1/(10^(db/10))/2);
   noiseb=noiseb*sqrt(1/(10^(db/10))/2);
   noise=sqrt(noisea.*noisea + noiseb.*noiseb);

   x(1)=exp(j*(2*pi*(fl(n/2000+1) - fc)*(n+o)/fs + theta)) +
noise(1)*exp(j*(2*pi*(fl(n/2000+1) - fc)*(n+o)/fs + phi1));

   x(2)=exp(j*(2*pi*(fl(n/2000+1) - fc)*(n+o+1)/fs + theta)) +
noise(2)*exp(j*(2*pi*(fl(n/2000+1) - fc)*(n+o+1)/fs + phi2));

   x(3)=0;
   x(4)=0;
   X=abs(fft(x));

   X24error(o/2+1)=X(2)*X(2)/4-X(4)*X(4)/4;
%    deltafTsAve(o/2+1) = 1/2/pi*real(asin(X(2)*X(2)/4 - X(4)*X(4)/4));
   clear x
   clear X
 end  %of o

 fl(n/2000+2)=fl(n/2000+1) - fs* 1/2/pi*real( asin(mean(X24error) ) );
```

```
  clear deltafTsAve;
  clear X24error
end  %of n

plot(fl/10^6)
pause
JF3mse(db/5+1)=1/length(fl(10:109))*((fl(10:109)-fc)*(fl(10:109)-fc)');


save JF3mse JF3mse

clear fl
clear noisea
clear phi2
clear db
clear fs
clear noiseb
clear theta
clear deltafTs
clear n
clear o
clear x
clear fc
clear noise
clear phi1
clear X

end  %end of db

db=0:5:50;
fc=100*10^6;
semilogy(db, sqrt(JF3mse)/fc*100, 'x')
hold on
semilogy(db, sqrt(JF3mse)/fc*100, ':')

!date

%
```

PERFORMANCES OF THE PHASE METHOD AFC AND THE MAGNITUDE METHOD AFC WITH FILTER 1, 2 , AND 3

```
clear
clf
figure(1)
load PFNmse
load PF1mse
load PF2mse
load PF3mse
fb=5*10^6;
db=0:5:50;
semilogy(db, sqrt(PFNmse)/fb*100, 'ow')
hold on
semilogy(db, sqrt(PFNmse)/fb*100, ':w')
semilogy(db, sqrt(PF1mse)/fb*100, 'ow')
semilogy(db, sqrt(PF1mse)/fb*100, ':w')
semilogy(db, sqrt(PF2mse)/fb*100, 'ow')
semilogy(db, sqrt(PF2mse)/fb*100, ':w')
semilogy(db, sqrt(PF3mse)/fb*100, 'ow')
semilogy(db, sqrt(PF3mse)/fb*100, ':w')
axis([-5 55 .005 100])
load JFNmse
load JF1mse
load JF2mse
load JF3mse
semilogy(db, sqrt(JFNmse)/fb*100, '+w')
hold on
semilogy(db, sqrt(JFNmse)/fb*100, ':w')
semilogy(db, sqrt(JF1mse)/fb*100, '+w')
semilogy(db, sqrt(JF1mse)/fb*100, ':w')
semilogy(db, sqrt(JF2mse)/fb*100, '+w')
semilogy(db, sqrt(JF2mse)/fb*100, ':w')
semilogy(db, sqrt(JF3mse)/fb*100, '+w')
semilogy(db, sqrt(JF3mse)/fb*100, ':w')
axis([-5 55 .005 100])
set(1, 'PaperPosition', [0 0 4.8 3.6])
```

ADAPTIVE AFC ALGORITHM

```
clear
for db=0:5:50,
db
L=4000;
LStart=L-1000;
fs=25*10^6;
fc=100*10^6;
fl(1)=98*10^6;
CB=5*10^6;
UBMax=102.5*10^6;
LBMin=97.5*10^6;
UB(1)=UBMax;
UB(2)=UB(1);
LB(1)=LBMin;
LB(2)=LB(1);
HitUp=0;
HitDown=0;
HitNone=0;
SM=5*10^6;
SMMax=5*10^6;
Expand=1.1;
Shrink=.9;

for n=0:2:L,
 theta=rand*2*pi;
 phi1=rand*2*pi;
 phi2=rand*2*pi;
 noisea=randn(1,2);
 noiseb=randn(1,2);
 noisea=noisea*sqrt(1/(10^(db/10))/2);
 noiseb=noiseb*sqrt(1/(10^(db/10))/2);
 noise=sqrt(noisea.*noisea + noiseb.*noiseb);

 fl(n+2)=fl(n+1);
 x(1)=exp(j*(2*pi*(fl(n+1) - fc)*n/fs + theta)) + noise(1)*exp(j*(2*pi*(fl(n+1) -
fc)*n/fs + phi1));
 x(2)=exp(j*(2*pi*(fl(n+2) - fc)*(n+1)/fs + theta)) + noise(2)*exp(j*(2*pi*(fl(n+2) -
fc)*(n+1)/fs + phi2));
 x(3)=0;
 x(4)=0;
```

```
X=abs(fft(x));

deltaf = 1/2/pi*fs*real(asin(X(2)*X(2)/4 - X(4)*X(4)/4));

fl(n+3)=fl(n+2) - deltaf;

if fl(n+3) >= UB(n+2)
  fl(n+3) = UB(n+2);
  HitUp = HitUp+1;
  HitDown = 0;
  HitNone = 0;
elseif fl(n+3) <= LB(n+2)
  fl(n+3)=LB(n+2);
  HitDown = HitDown+1;
  HitUp = 0;
  HitNone = 0;
else
  HitUp = 0;
  HitDown = 0;
  HitNone = HitNone+1;
end

if (HitUp == 10) | (HitDown == 10)
  SM=SM * Expand;
  HitUp = 0;
  HitDown = 0;
  HitNone = 0;

elseif HitNone == 1
  HitUp = 0;
  HitDown = 0;
  HitNone = 0;
  SM = SM * Shrink;
end

if SM > SMMax
  SM=SMMax;
end
UB(n+3)=fl(n+3) + SM;
LB(n+3)=fl(n+3) - SM;
if UB(n+3) > UBMax
  UB(n+3) = UBMax;
end
if LB(n+3) < LBMin
```

```
  LB(n+3) = LBMin;
 end

 UB(n+4)=UB(n+3);
 LB(n+4)=LB(n+3);

end  %of n

adaptivemse(db/5+1)=1/length(fl(LStart:L))*((fl(LStart:L)-fc)*(fl(LStart:L)-fc)');

figure(1)
plot(fl/10^6)
figure(2)
clf
subplot(211)
qq=1:100;
plot(fl(qq)/10^6)
ylabel('Fl, MHz')
hold on
plot(UB(qq)/10^6,'g')
plot(LB(qq)/10^6,'r')
axis([1 100 min(LB(qq)/10^6) max(UB(qq)/10^6)])
subplot(212)
plot((UB(qq)-LB(qq))/10^6)
ylabel('Estimation band, MHz')
figure(3)
clf
subplot(211)
plot(LStart:L, fl(LStart:L)/10^6)
hold on
plot(LStart:L, UB(LStart:L)/10^6, 'g')
plot(LStart:L,LB(LStart:L)/10^6, 'r')
ylabel('FL, MHz')
axis([LStart L min(LB(LStart:L)/10^6) max(UB(LStart:L)/10^6)])
subplot(212)
stairs(LStart:L, UB(LStart:L)/1000 - LB(LStart:L)/1000)
ylabel('Estimation band, KHz')

db
pause
clear fl
clear UB
clear LB
end  %of db
```

```
save adaptivemse adaptivemse
db=0:5:50;
figure(4)
load JFNmse
semilogy(db, sqrt(JFNmse)/CB*100, 'yo')
hold on
semilogy(db, sqrt(JFNmse)/CB*100, ':')
semilogy(db, sqrt(adaptivemse)/CB*100, 'rx')
semilogy(db, sqrt(adaptivemse)/CB*100, '--')
axis([0 50 .01 100])
%
```

PERFORMANCES OF THE MAGNITUDE METHOD AFC AND THE

ADAPTIVE AFC WITH DIFFERENT CONFIGURATIONS

```
clear
clf
figure(1)
fb=5*10^6;
db=0:5:50;
load JFNmse
load adaptive1a
load adaptive2a
load adaptive3a

semilogy(db, sqrt(JFNmse)/fb*100, '+w')
hold on
semilogy(db, sqrt(JFNmse)/fb*100, '-.w')
semilogy(db, sqrt(adaptive1a)/fb*100, '*w')
semilogy(db, sqrt(adaptive1a)/fb*100, '-.w')
semilogy(db, sqrt(adaptive2a)/fb*100, 'ow')
semilogy(db, sqrt(adaptive2a)/fb*100, '-.w')
semilogy(db, sqrt(adaptive3a)/fb*100, 'xw')
semilogy(db, sqrt(adaptive3a)/fb*100, '-.w')
axis([-5 55 .005 100])
set(1, 'PaperPosition', [0 0 4.8 3.6])
```